TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

NÁVRH EXPERTNÍHO SYSTÉMU PRO ANALÝZU SPOTŘEBY ELEKTRICKÉ ENERGIE

Bakalářská práce

Studijní program: B2612 – Elektrotechnika a informatika
Studijní obor: 1802R022 – Informatika a logistika

Autor práce: **Erik Mikula**
Vedoucí práce: Ing. Jan Kraus, Ph.D.

Liberec 2014

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

DESIGN OF AN EXPERT SYSTEM FOR THE ANALYSIS OF ENERGY CONSUMPTION

Bachelor thesis

Study programme: B2612 – Electrical Engineering and Informatics
Study branch: 1802R022 – Informatics and Logistics

Author: **Erik Mikula**
Supervisor: Ing. Jan Kraus, Ph.D.

Liberec 2014

Tento list nahraďte
originálem zadání.

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Poděkování

Tímto děkuji vedoucímu práce Ing. Janu Krausovi, Ph.D. za poskytnutí reálných dat k analýze. Také chci poděkovat Ing. Jiřímu Kubínovi, Ph.D. za konzultaci.

Abstrakt

Práce se zabývá vývojem aplikace pro analýzu rozsáhlých datových souborů z měření elektrických charakteristik tramvají. Jsou použity open-source nástroje – programovací jazyk Python a knihovny NumPy, Pandas, Matplotlib, Scikit-learn. Je popsána oblast použití knihovny Pandas, a to jak její silné stránky, tak i implementační omezení. Předmětem diskuse je, s jakými výzvami se v současnosti potýkají datoví analytici. Část se věnuje potenciálním procesním rizikům při vývoji softwarových systémů tohoto typu. V dokumentaci navrženého systému je zdůvodněna navržená architektura a rozebrány případy užití jednotlivých typů vizualizací realizovaných pomocí knihovny Matplotlib. Prostor je rovněž věnován podoboru umělé inteligence – učení bez učitele. Možnosti knihovny Scikit-learn byly prověřeny během srovnání vybraných shlukovacích algoritmů, obsažených v knihovně. Vybraný shlukovací algoritmus je použit při detekci provozních režimů elektrické trakce tramvaje a při hledání typických stavů na agregovaných datech.

Klíčová slova:

data, Big data, analýza dat, časové řady, explorační analýza, Python, Numpy, Matplotlib, Pandas, Scikit-learn, K-means, AffinityPropagation, DBSCAN, MeanShift

Abstract

The present thesis aims at the development of application for analysis of extensive data files that cover measurements of electrical characteristics of tramways. Open-source tools are used – programming language Python and libraries NumPy, Pandas, Matplotlib, Scikit-learn. The application field of the Pandas library is described – both its strong points and its limitations of implementation. The discussion covers the current challenges that the data analysts have to deal with. Part of it aims at potential process risks when developing software systems of this type. Proposed architecture is rationalized in the documentation of the designed system and cases of application of individual visualization types realized by the Matplotlib library are analysed. The artificial inteligence subarea - specifically learning without a teacher – is also covered to a certain extent. The possibilities of the Scikit-learn library were tested when comparing of the selected cluster algorithms from the library took place. Selected cluster algorithm is applied during detection of operating modes of electrical traction of tramways and during the search for typical states on the aggregated data.

Key words:

data, Big data, data analysis, time series, exploratory analysis, Python, Numpy, Matplotlib, Pandas, Scikit-learn, K-means, AffinityPropagation, DBSCAN, MeanShift

Obsah

Seznam grafů

Seznam obrázků

1 Úvod

V této kapitole informuji o základních pojmech, se kterými se pravděpodobně setká každý zájemce o datovou analýzu. Naznačuji vývoj strojového zpracování dat od minulosti, přes současnost, až po blízkou budoucnost. Na to navazuji specifikací mnou řešeného úkolu.

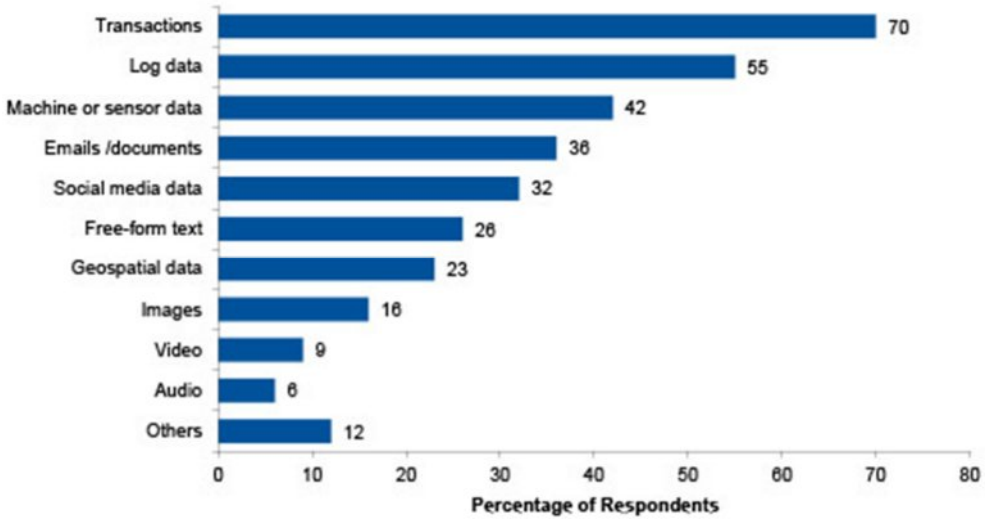
1.1 Kontext zpracovávaného problému

Statistické zpracování úplného souboru dat bylo vždy nákladnější a obtížněji realizovatelné, než zpracování výběrového vzorku. Již v roce 1890 bylo v USA provedeno sčítání lidu pomocí strojů zpracovávajících děrné štítky (1). Fenoménem posledních desetiletí je enormní zvyšování množství dat, kterým lidstvo disponuje. Současně s růstem výkonu počítačových systémů a růstem kapacity jejich paměťových subsystémů roste i množství strojově zpracovatelných dat. Datový průmysl, výzkum v různých aplikačních doménách, ale i management firem si klade otázku, zda by shromážděná data mohla být užitečná. Užitečnost dat by měla spočívat v možnosti činit lepší rozhodnutí na základě informací poskytnutých analytickým systémem. Podle výzkumu, který provedli ekonom Erik Brynjolfsson a jeho kolegové z MIT a Penn's Wharton School se rozhodovací procesy řízené daty[1] projevují příznivě na výkonnosti firmy, což je v pozitivní korelaci s tržní hodnotou akcií (2).

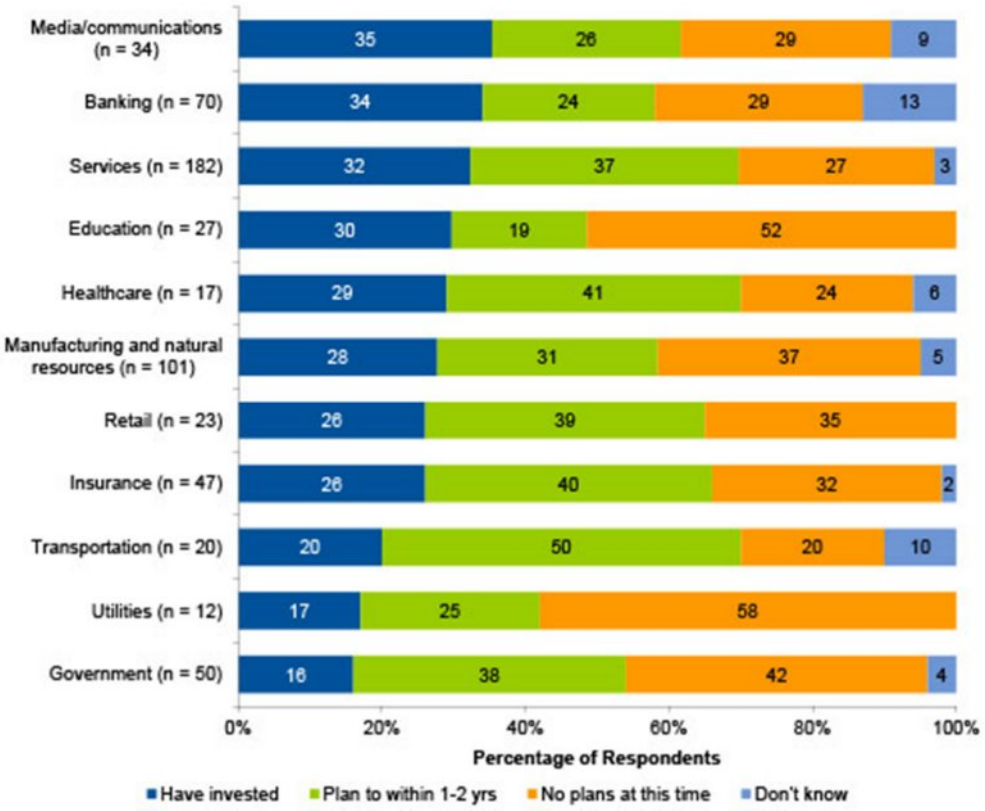
1.2 Big data

Trendem posledních let je řešení tzv. Big data problémů[2]. Podle někoho jsou Big data pouze marketingem zprofanovaný prázdný pojem, podle jiných jde o strategickou investici. Ať již na to máme jakýkoli názor, faktem je, že trend investic do budování Big data infrastruktury je rostoucí (viz Graf 2). Analyzují se data v řadě odvětví, např.: výrobní procesy, bankovnictví, medicína, digitální média, chování zákazníků a poslední dobou se tento trend projevuje i v poměrně konzervativním oboru – v energetice (tzv. smart grid, neboli inteligentní rozvodné sítě).

[1] V originálu: data-driven decision-making.
[2] Big data je takové množství dat, že je nelze analyzovat konvenčními prostředky.



Graf 1 - typy dat, které bývají nejčastěji analyzovány (22).



Graf 2 - investice do vývoje Big data infrastruktury podle odvětví (22).

Skutečností však zůstává, že podstatná část generovaných dat rychle zastarává a informace v nich obsažené již po nějakém čase nemusí být aktuální a tedy ani hodnotné. Miko Matsumura srovnává Big data s hnijící velrybou (3).

Obrázek 1 - žádná z vizualizací na této tabuli nedává smysl (3).

Podstatou práce s daty by měl být skeptický přístup k výsledkům a zaměření více na kontext, než jenom na práci s čísly (4).

„Nalezené odchylky mohou plynout buď z nepřesností při sběru a záznamu dat, nebo mohou být indikátorem skutečných odchylek oproti standardnímu průběhu.“ (5)

1.3 Strukturovaná data

Aby bylo možné data zpracovávat, musí být k dispozici ve strukturované podobě. Pro transformaci z („surových“, „špinavých“) nestrukturovaných dat do datové struktury vhodné k dalšímu zpracování se rozšířil anglický slangový výraz munge / munging / popř. wrangling, pro který jsem nenašel přímý český ekvivalent. Pojem se do jisté míry překrývá se zkratkou ETL[3], používanou zejména v oblasti podnikových informačních systémů pro podnikové procesy, pří kterých jsou pomocí specializovaného software data převáděna z různých provozních systémů do jednoho datového skladu. Strukturovaná data mohou mít podobu (6):

- Multidimenzionální pole (matice).
- Tabulková data – sloupce mohou mít různé datové typy, mnohdy doplněné meta-daty (názvy sloupců, atp.). Např. csv soubor, nebo tabulka relační databáze.
- Víc tabulek propojených přes klíčové sloupce.
- Rovnoměrně nebo nerovnoměrně rozmístěné časové řady.

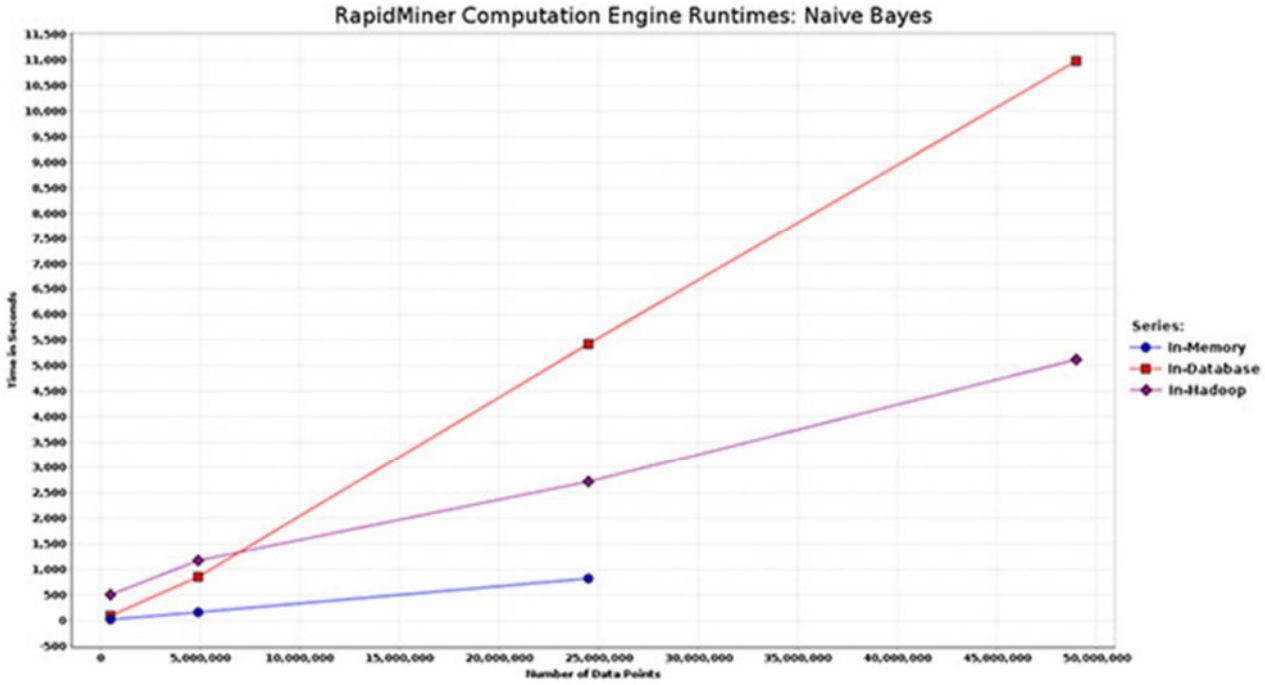
1.4 Způsoby zpracování dat

Zpracování dat lze provádět třemi způsoby:

[3] Extract, transfer, load

- in-memory
- in-database
- distribuovaně (např. Apache Hadoop, nebo Apache Spark)

Zpracování in-memory (v paměti) je nejrychlejší, ale kapacita RAM výpočetního systému nemusí být dostatečná pro všechny typy úloh. Při zpracování v databázi sice není prakticky omezena velikost datasetu[4], ale rychlost nemusí být dostatečná. Pokud se chceme analýzy většího množství dat dočkat v přijatelném čase, může být jediné použitelné řešení v nasazení distribuované architektury.



Graf 3 - závislost doby zpracování paralelizovatelné úlohy s lineární časovou složitostí na počtu datových bodů (21).

1.5 Expertní systémy

Expertní systém je software, který simuluje rozhodovací činnost experta při řešení složitých úloh (7). Problematikou tvorby expertních (znalostních) systémů se zabývá znalostní inženýrství. Na vstupu je popis problému. Poté inferenční mechanismus pomocí báze znalostí vyvodí závěry. Báze znalostí může být získána (7):

- od expertů formou spolupráce mezi znalostními inženýry a experty (1:1, 1:n, m:1, m:n)
- automatizovaně pomocí strojového učení (od expertů, z textů, z dat)

[4] Soubor zkoumaných dat.

Hotový systém je schopen činit kvalifikovaná rozhodnutí, což by mělo ušetřit náklady na živého experta. Výstupem může být např. upozornění na nestandardní stavy ve sledovaném systému, na které je třeba reagovat, aby nedošlo k nežádoucím dopadům na sledovaný systém.

1.5.1 Strojové učení

Strojové učení je oblast umělé inteligence, která má za cíl umožnit počítačovému systému přizpůsobit se okolnímu prostředí, což bývá výhodné při řešení rozhodovacích problémů. V podstatě se jedná o praktickou aplikaci teoretických poznatků z pravděpodobnosti, statistiky a dalších oblastí matematiky. Rozlišujeme (zdroj: materiály k předmětu ITE/MRK):

- Učení s učitelem – učení se z označkovaných dat.
- Učení bez učitele – hledání skryté struktury neoznačkovaných dat.
- Zpětnovazební učení – systém se soustavně učí za provozu na základě důsledků jím provedených akcí.

1.6 Datové sklady

Datové sklady jsou na trhu zavedeným artiklem již delší dobu. Jejich úkolem je poskytnout vhodné úložiště pro obsáhlé datové soubory, určené kromě archivace zejména k další analýze (nějaký druh OLAPu[5]).

1.7 Specifikace řešeného problému

Bakalářská práce zpracovává datové soubory, získané při měření na elektrických subsystémech tramvají v liberecké dopravní síti. V datovém skladu Envis jsou uloženy v silně komprimované podobě. Vstupem pro mé analýzy byly dekomprimované csv soubory o velikosti řádově jednotek Gigabajtů. V jednom takovém souboru jsou uloženy údaje z měření na jedné tramvaji po dobu několika dní. Při nasazení na více linkách, ve více městech, po větší část roku by snaha o zpracování dat z měření mohla přerůst do Big data problému. Nicméně objemy zpracovávané v této práci nepřerostly rozsah standardního datového skladu.

1.8 Cíle práce

Cílem práce bylo pomocí vybraných technologií realizovat aplikaci, která bude provádět explorační a shlukovou analýzu, což by mělo posloužit jako základ pro tvorbu Expertního systému v dalších etapách. Explorační analýza se zabývá souhrnnými charakteristikami a vizualizací. Umožňuje získat představu o chování sledovaných

[5] Online analytical processing.

veličin a stanovit metody vhodné pro bližší zkoumání (5). Cílem shlukové analýzy je zjištění, zda v datech existují kategorie prvků odlišných vlastností (5). Po zvládnutí úvodních etap by měla být provedena akvizice znalostí od experta na oblast elektrické energie a získané znalosti následně využity při tvorbě pokročilejších analýz.

Dílčím úkolem bylo prověřit, zdali by z těchto dat šlo získat zajímavé, doposud neznámé souvislosti, které by potenciálně mohly zvýšit výkonnost libereckého dopravního podniku.

Přínosem pro objednatele by měla být úspora nákladů na lidské zdroje při zpracování výstupů z měření. Přínosem pro autora práce by mělo být porozumění technologiím zpracování dat z oblasti open-source.

2 Postup při realizaci

Stejně jako u jakéhokoliv jiného druhu projektu by i vypracování bakalářské práce mělo probíhat pomocí nějaké systematické metodiky, vedoucí k vytyčenému cíli. Tato kapitola popisuje problémy procesního charakteru, které musely být vyřešeny. Protože se výběr použitých technologií neobešel bez komplikací, byla tato část softwarového procesu také zařazena do této kapitoly.

2.1 Analýza rizik projektu

Práce na projektu začala vyhotovením předběžné analýzy rizik ve formě strukturované FMECA analýzy. Zpětně se dá konstatovat, že řada rizik byla identifikována správně, avšak realizovaná opatření jak jim předejít, byla v několika případech nedostatečná.

Tabulka 1 – FMECA analýza.

Riziková situace	Pst.	Dopad*	Míra rizika	Řešení
Nedodržení domluveného rozsahu v termínu odevzdání.	50%	9	4,5	Více pracovat, nebrat zpracovávaný problém na lehkou váhu. Volba priorit.
Nedostatečné, nebo nerealistické korekce stanovených milníků projektu vedoucím pracovníkem.	40%	7	2,8	Nepodceňovat význam komunikace jednotlivých aktérů.
Slabá motivace zpracovatele.	40%	7	2,8	Osobní zainteresovanost na co nejlepším výsledku.
Nedostatky v administrativní části projektu	50%	5	2,5	Zpracovatel, i vedoucí budou sledovat instrukce na webu fakulty.
Zadavatel nepopíše dostatečně své požadavky.	20%	9	1,8	Systematická analýza požadavků.
Nedostatečná znalost použitých technologií zpracovatelem.	20%	7	1,4	Studium kvalitní literatury odpovídajícího rozsahu + praktické cvičení.
Dlouhodobá pracovní neschopnost zpracovatele.	20%	7	1,4	Vytvoření časové rezervy.
Ztráta dat způsobená kolapsem HW, SW, nebo útokem malware.	15%	9	1,35	Zálohování dat. Kvalifikovaná správa pracovní stanice.

Aplikace obsahuje mnoho chyb.	20%	5	1	Kvalitní návrh architektury a následná verifikace implementovaných částí.
Rozpor mezi požadavky zadavatele a zkušební komise.	10%	9	0,9	Zpracovatel si přečte posudky oponentů z minulých let.
Neodkladné časové zaneprázdnění zpracovatele záležitostmi s vyšší prioritou.	30%	3	0,9	Vytvoření časové rezervy.

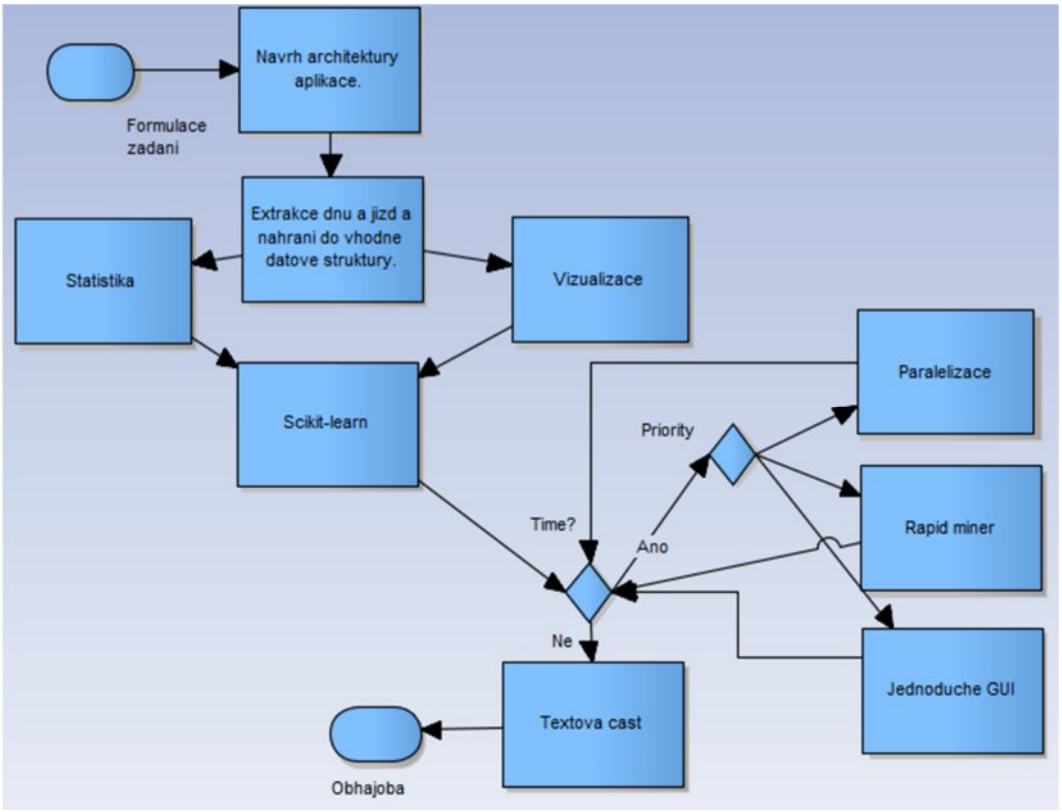
*** Dopad: 1 – zanedbatelný, 3 – nízký, 5 – střední, 7 – vážný, 9 – kritický**

Na základě znalostí a zkušeností, kterými jsem disponoval v přípravné fázi, došlo k přecenění role objektového návrhu architektury aplikace. V situaci, kdy je specifikace zadání vágní a lidské zdroje omezené, je lepší nejprve vytvořit funkční prototypy částí aplikace a až když je funkcionalita na akceptovatelné úrovni, začít přemýšlet o tom, jak dílčí zdrojové kódy integrovat do jednoho udržovatelného a rozšiřitelného celku.

2.2 Harmonogram prací

Prvním krokem procesu zpracování bakalářské práce bylo určení rámcového harmonogramu prací (viz Obrázek 2). Jak bylo CASE nástrojem Enterprise architect doporučeno, byl tento harmonogram dán na schválení vedoucímu práce.

V době psaní těchto řádků již je zřejmé, že cíle byly stanoveny nerealisticky. Stihl jsem realizovat funkční prototyp aplikace podle vstupní – „povinné“ větve. Volitelné větve: paralelizace, referenční řešení pomocí Rapid mineru, jednoduché GUI pro vstupy od uživatele by rozsahem stačily na další jednu až dvě bakalářské práce. Protože konkrétní funkcionalita nebyla zadáním specifikována, nelze jednoznačně vyhodnotit, zdali je rozsah praktické části vyšší, či nižší, než bylo požadováno.



Obrázek 2 - plánovaný harmonogram prací na projektu Bakalářská práce.

2.3 Volba technologií

Ve věci volby technologií jsem měl vysokou míru volnosti. Během úvodních jednání o zadání mi bylo doporučeno použít skriptovací programovací jazyk Python, pro který je k dispozici řada knihoven pro práci s daty. Konkrétně mi bylo doporučeno použít Matplotlib a Scikit-learn.

Při počátečním vyhledávání vhodných nástrojů jsem našel knihu Python for Data Analysis (6), která popisovala použití knihoven NumPy, Matplotlib a zejména Pandas. Zaměření knihy odpovídalo zpracovávanému problému. Bohužel jsem se tolik soustředil na objektový návrh architektury aplikace, že kniha zůstala zpočátku založena mezi větším množstvím výsledků vyhledávání. Vedoucí po mě chtěl rychle dílčí výsledky a já postupně realizoval něco, co funkcionalitou vzdáleně připomínalo zmíněnou knihovnu Pandas, s rozdílem výrazně menší funkcionality i výkonu.

2.4 Metodika vývoje

Po několika měsících, kdy jsem vyvíjel něco, co již bylo vymyšleno a z požadovaných funkcí bylo implementováno jen málo, jsem se rozhodl začít úplně od začátku, tentokrát s pomocí knihovny Pandas. Napjatou situaci jsem zkusil vyřešit odlišným přístupem k vývoji. Začal jsem se nejdříve učit používat vybrané knihovny

a pomocí nich implementovat funkcionalitu v jediném zdrojovém souboru, téměř bez použití OOP. Tím jsem se během několika dní dostal na úroveň funkcionality, kterou jsem předtím programoval desítky až stovky hodin. Jak rozsah zdrojového kódu rostl, přestával být přehledný. Proto jsem strukturu aplikace, volně vycházející z návrhového vzoru MVC, přeci jenom vytvořil (podrobněji v 3.1). Tentokrát jsem však nedodržoval tak striktně principy objektového návrhu[6], vedoucí k rozšiřitelným, udržovatelným a spolehlivým aplikacím. Vlastně jsem vyměnil snadnou rozšiřitelnost za funkcionalitu. Tímto způsobem jsem implementoval explorační analýzu. Času mnoho nezbývalo a já jsem chtěl za každou cenu získat přehled o shlukovacích algoritmech, obsažených v balíku scikit-learn. Vyexportoval jsem některé číselné charakteristiky, potenciálně vhodné jako příznakové vektory, do souborů ve formátu NumPy polí (viz 2.6). Nad nimi jsem pak prováděl adhoc analýzy pomocí sady skriptů (co algoritmus, to skript). Takto jsem vytipoval vhodný algoritmus, který jsem integroval do aplikace.

2.5 Python

Před začátkem řešení bakalářské práce jsem se setkal již s několika programovacími jazyky se syntaxí vycházející z C. Nicméně Python byl pro mne novinkou a zároveň příjemným překvapením. Byl vytvořen v roce 1991 a pro vědeckotechnické výpočty v průmyslu i akademické sféře se začal používat krátce po roce 2000 (6). Jedná se o platformně nezávislý, vysokoúrovňový skriptovací open-sourcový jazyk, s velikým množstvím dostupných knihoven a efektivní syntaxí. Při psaní zdrojového kódu jsou k dispozici pokročilé příkazy a datové struktury, zabudované do syntaxe jazyka tak, že je jejich použití snadné a není třeba popsat tolik řádků zdrojového kódu[7], jako ve většině jiných programovacích jazyků. Umožňuje to soustředit se více na řešený problém. V komunitě[8] okolo zpracování dat se navíc těší vysoké popularitě.

Při průmyslovém řešení datových problémů se často použije některý z doménově specifických jazyků, jako jsou R, MATLAB, SPSS Modeler, RapidMiner pro rychlou tvorbu prototypu. Produkční prostředí je poté vyvíjeno např. v C#, Javě, nebo C++. Se vznikem knihovny Pandas (viz 2.8) se Python stal silnou alternativou pro datové úlohy. Spojuje výhody rychlého vývoje v doménově specifických jazycích s možnostmi obecných programovacích jazyků.

[6] Např. princip re-use.
[7] Podle autora (6) umožňuje vytvářet „quick and dirty small programs“
[8] Důkazem může být množství příspěvků na konferenci Strata pořádané nakladatelstvím O'Reilly.

Bylo by možné oponovat, že interpretované jazyky jsou v řadě situací pomalejší oproti jazykům kompilovaným. Ve vědecko-technických výpočtech, kam patří i datově intenzivní aplikace, však Python slouží jako „lepidlo“, které se vykonává jen malé procento z celkového času běhu programu. Výpočetně náročnější části kódu jsou implementovány pomocí knihoven napsaných např. v C, C++, nebo fortranu (6). U zrodu tohoto přístupu stála idea, že programátorův čas je cennější, než výpočetní čas procesorové jednotky. Výjimkou mohou být aplikace, ve kterých je kritickým aspektem extrémně rychlá latence, jako je vysokofrekvenční obchodování na burze. V takovém případě jdou náklady na programátorský tým stranou a implementace probíhá v C++.

Slabinou by mohl být GIL [9], který je typický pro skriptovací jazyky. Mechanismus GIL neumožňuje používat víc souběžně spuštěných vláken v rámci jednoho spuštěného procesu – interpreteru. Knihovny pro meziprocesovou a meziprocesorovou komunikaci však existují, proto je možné použít Python i pro Big data.

Jistá nevýhoda vyplývá z toho, že to není jazyk původně určený pro rozsáhlejší aplikace čítající desítky, stovky, či více tříd a proto je hůře podporován v CASE nástrojích. Zkoušel jsem celou řadu nástrojů pro generování kostry aplikace z UML Class diagramu, nebo naopak pro generování diagramu z existujícího projektu a musím konstatovat, že jazyky jako C#, nebo Java jsou v tomto aspektu dále.

2.6 NumPy

NumPy je knihovna napsaná v jazyce C, která zjednodušuje a zefektivňuje práci s vícerozměrnými poli v jazyce Python. Jedná se o jeden z modulů balíku SciPy který dále obsahuje moduly: *„optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.“* (8)

Základem jsou rychlá multidimenzionální pole, reprezentovaná pomocí objektů typu ndarray. Dále je v modulu obsaženo (6):

- Metody pro výpočty uvnitř pole, či mezi více instancemi polí.
- Čtení / zápis na HDD.
- Generátor pseudonáhodných čísel.
- Nástroje pro integraci s C, C++, Fortranem.

[9] Global interpreter lock

2.7 Matplotlib

Knihovna pro 2D i 3D vizualizace, jejíž možnosti i ovládací rozhraní jsou srovnatelné s vizualizacemi v Matlabu (jak bude vidět v kapitolách 3 a 4). Podporuje všechny běžné operační systémy a umožňuje export do formátu PNG i do vektorového SVG.

2.8 Pandas

Knihovna, která je vyvíjena od roku 2008, vznikla původně za účelem snadné analýzy dat z finančních trhů. V současné době se hodí pro in-memory zpracování libovolných časových řad. V podstatě se jedná o vysokoúrovňovou nadstavbu knihovny NumPy, doplněnou o řadu funkcí. Hlavní nevýhodou je, že prozatím není k dispozici nativní podpora paralelního zpracování. Mezi výhodami je třeba zmínit, že se jedná o open-source, kolem kterého je aktivní komunita a proto se stále vyvíjí. Vznikl jako obdoba statistického software R, použitelná v Pythonu. Z hlediska možností a snadnosti používání se jedná o zajímavou alternativu ke statististickým výpočtům v prostředí MATLAB (který se však obtížněji integruje do informačních a řídicích systémů a je to komerční software s nezanedbatelnou cenou licence).

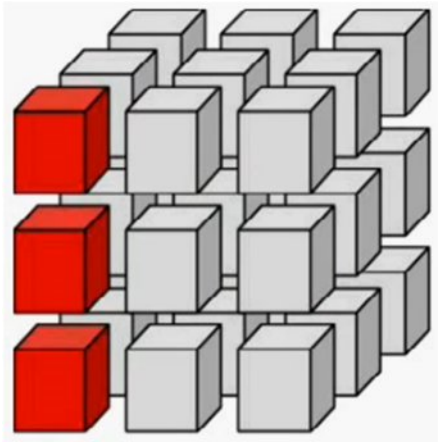
Autor knihovny je zároveň autorem knihy (6), ve které popisuje důvody, které ho vedly k vytvoření knihovny. Chtěl mít k dispozici snadno použitelnou knihovnu pro:

Interakci s okolním světem.

- Čtení a zápis široké škály formátů a databází.

Přípravu dat.

- Čištění – ošetření nekorektních, nebo chybějících hodnot. Zahrnuje převody datových typů, parsování datumů, reprezentace formátu čísel s plovoucí desetinnou čárkou (čárka vs. tečka). U chybějících hodnot je možnost dopředné, či zpětné propagace, interpolace, atp.
- Slučování více datasetů, včetně zarovnání na společný index (data alignment).
- Transformace tvaru datové struktury.
- Slice and Dice – pojmy z operací nad OLAP kostkou. Slice představuje příčný řez. Výstupem je struktura s dimenzí o jednu nižší. Dice vrací podmnožinu původní kostky při zachování původní dimenze prostoru. Více v (9).
- Další přípravné operace (např. změny vzorkovací periody nahoru i dolů)



Obrázek 3 – trojrozměrná datová struktura, po dvou řezech (slice) (23).

Transformaci

- Provádění matematických a statistických operací nad skupinami dat, k odvození nových datasetů. Například agregace velké tabulky na skupinu proměnných.

Modelování a výpočty

- Napojení dat na statistické modely, algoritmy strojového učení a jiné výpočetní nástroje.

Prezentace

- Vytváření interaktivních nebo statických vizualizací, případně textových souhrnů.

Základní datovou strukturou je 1D série (varianta s časovým, nebo pořadovým indexem). Ze sérií se skládají 2D DataFrame, které jsou obdobou data.frame z prostředí R. V případě potřeby je možné DataFrame seskupovat do 3D kontejneru nazvaného Panel.

3 Popis implementace

V této kapitole nejprve píšu o technickém řešení a zdůvodňuji, proč je právě takové, jaké je. Poté popisuji případy užití jednotlivých druhů vizualizací. Zvídavý čtenář si mnou implementované algoritmy jistě prohlédne přímo ve zdrojových kódech na přiloženém DVD.

3.1 Architektura

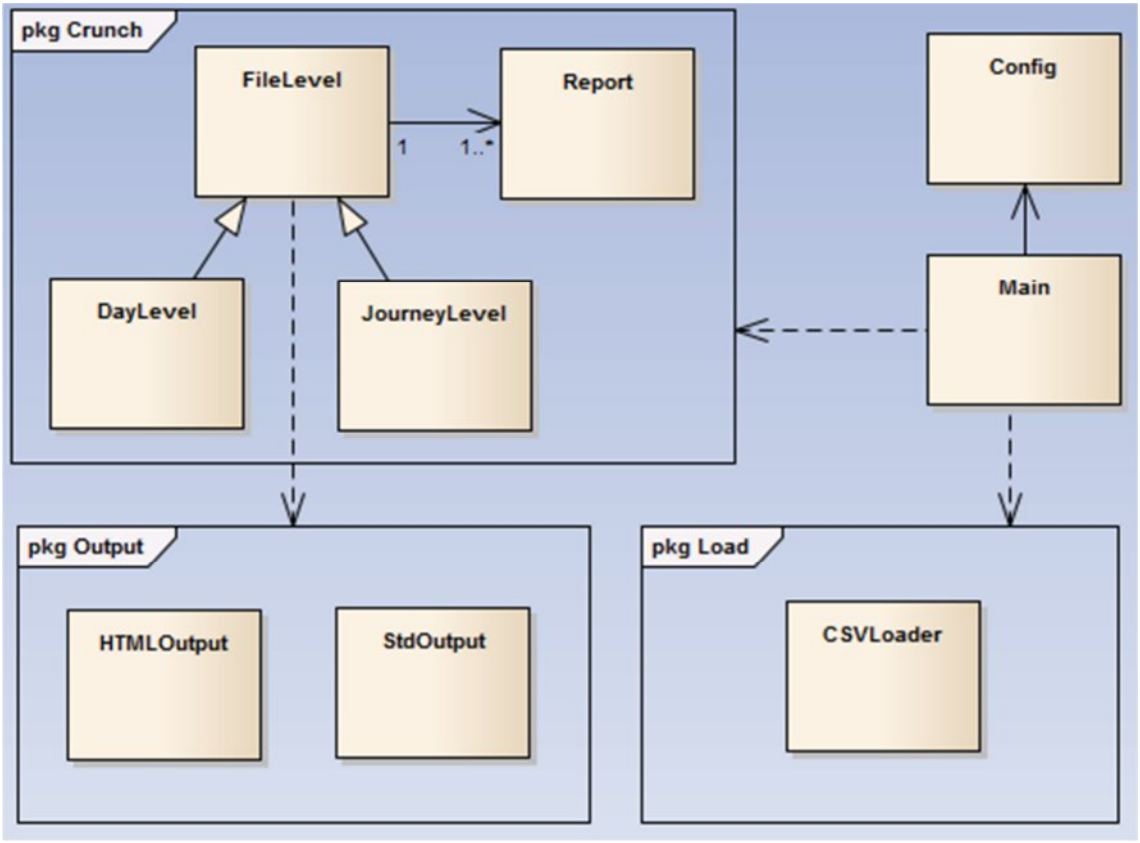
Myšlenka návrhu je taková, že vstupy od uživatele jsou přijímány pomocí nastavení konstant ve zdrojovém souboru Config.py. Soubor Main.py tvoří „controller“, kdy podle těchto vstupů vytvoří instance odpovídajících tříd a zavolá metody těchto instancí s odpovídajícími parametry. Většinu výpočetního času spotřebuje přípravná fáze zpracování, kterou zajišťuje třída CSVLoader a pak samotná analýza prováděná ve třídách z balíčku Crunch[10]. Návrh počítá se se třemi úrovněmi agregace dat.

Úroveň agregace celého souboru tvoří v aktuální verzi spíše funkci rodičovské třídy pro nižší úrovně agregace. Umí vrátit množinu unikátních dní v datovém souboru. Kromě toho implementuje společnou funkcionalitu pro všechny úrovně agregace, což je nastavení reference na instanci třídy Report (strukturální návrhový vzor sdílená agregace, pozor neplést s agregací v datech; více v (10)), který je po naplnění požadovanými konfigurovatelnými datovými výstupy (zvolené grafy, a textové shrnutí) předán všem nastaveným instancím implementujícím rozhraní Reporter z balíčku Output.

V úrovni agregace na dny jsem implementoval funkcionalitu, kterou pan Ing. Jiří Kubín, Ph.D. požadoval (více v Kapitole 3.4).

Úroveň agregace na jednotlivé jízdy je v základu okopírovaná úroveň na dny, nicméně doplněná o další funkce. Zde byl porušen princip návrhu re-use, nicméně jak jsem již vysvětlil, vyšší prioritu jsem přidělil rozšíření funkcionality. Skutečný stav zdrojového kódu v této třídě připomíná spíše jakési „abstraktní staveniště“ (více v Kapitole 3.5).

[10] Jedná se o zaužívaný slangový název v doslovném překladu znamenající „chroustat“.



Obrázek 4 – UML diagram, zobrazuje zjednodušenou strukturu aplikace.

3.2 Příprava dat

Načtení do operační paměti a jejich příprava pro další zpracování probíhá ve třídě CSVLoader. Při dodržení odpovídajícího rozhraní by bylo možné aplikaci rozšířit o další typy vstupů bez nutnosti rozsáhlých úprav aplikace. V této třídě jsem pomocí výjimek ošetřil standardní chybové stavy, vyskytující se u souborově orientovaných vstupů. Dále je implementováno:

- Detekce vzorkovací periody s přesností na jednotky milisekund.
- Výběr sloupců ze vstupního souboru.
- Určení datového typu jednotlivých sloupců.
- Parsování datumu a času jednotlivých vzorků.
- Určení oddělovače v csv souboru na vstupu.
- Určení, zda se na vstupu používá desetinná čárka, nebo tečka.
- Výběr počtu vzorků ze vstupního souboru, které se mají načíst.
- Test, zda je v celém souboru použita stejná vzorkovací perioda. Při dalším zpracování se to předpokládá, protože to urychluje výpočty.
- Metoda pro návrat názvu vstupního souboru.

To, čemu se v terminologii dataminingu[11] říká příprava dat, je provedeno v režii funkce read_csv(), knihovny Pandas, která je volána v konstruktoru třídy CSVLoader. Ač funkce podporuje automatickou detekci formátu času a data, explicitní zadání formátu funguje řádově rychleji.

Výstupem je standardní datová struktura Pandas.DataFrame (návrhový vzor Factory). Drobné zdržení při vývoji způsobil fakt, že jsem měl k dispozici jednak data se vzorkovací periodou 200 ms a jednak 5 s, přičemž formát času je v těchto případech odlišný. V závislosti na operacích, prováděných v dalších fázích zpracování, se přesná podoba datové struktury může lišit v počtu sloupců (viz pojem transformace v 2.8). Základní podoba vypadá jako v tabulce (viz Tabulka 2). Duplicita data je v tomto případě záměrná – oba sloupce budou dále využívány, přičemž nebude potřeba spotřebovávat procesorový čas na opakované konverze. Při agregaci na dny (viz 3.4), bude podle sloupce date provedena stratifikace datasetu a bude použit jako nový index. Jako výchozí indexování je použito číslo řádku.

```
columnsUIP = ["record time[s]","avg.U1[V]", "avg.IL1[A]", "avg.P1[kW]"]
fmt = '%m/%d/%Y %I:%M:%S %p'
self.df = pd.read_csv(INPUT_FILE_PATH, delimiter=';', header=1, skiprows=0,
                      usecols=columnsUIP, decimal = ",",
                      dtype = {"avg.U1[V]": np.float64,
                               "avg.IL1[A]": np.float64, "avg.P1[kW]": np.float64},
                      nrows=NSAMPLES,
                      date_parser = lambda d: datetime.datetime.strptime(d, fmt),
                      parse_dates=["record time[s]"] )
```

Fragment kódu 1 - podstatnou část přípravy dat vykoná funkce Pandas.read_csv()

Tabulka 2 – základní podoba uložených dat v DataFrame

[index]	record time[s]	avg.U1[V]	avg.IL1[A]	avg.P1[kW]	date
0	8.11.2013 10:47:25	696.150	26.336	1.271635	8.11.2013
...	...	...	...	...	...
n	...	...	...	...	...

Třída CSVLoader si zcela bez problému poradí se souborem o velikosti 1.5GB (1936320 datových bodů) na počítači, který je vybaven 4GB RAM. Rezerva ve velikosti podporovaných souborů zde stále je. Pokud by bylo dosaženo stropu, je možné použít při načítání „kouskovač“ anglicky chunker. Tím lze soubor zpracovávat po blocích, mezi kterými lze iterovat. Pokud bychom narazili na limity souborového

[11] Datamining je statistické zpracování dat třetí a čtvrté generace (5).

systému, popsaná funkce podporuje HDF5, pro který existují i distribuované implementace (možnost paralelního přístupu k souborovému systému).

Limitujícím faktorem velikosti vstupu je tedy velikost dostupné operační paměti. U větších souborů a podrobnějších analýz (viz 3.4 a 3.5) je pak úzkým hrdlem skutečnost, že Pandas nativně nepodporuje paralelní zpracování. Na adrese *github.com/pydata/pandas/issues/5751* probíhá diskuze vývojářů projektu o způsobu implementace paralelismu. Do budoucna se plánuje podpora paralelního běhu na systémech s více výpočetními jádry, se sdílenou pamětí.

Z toho vyvozuji závěr, že jde při odpovídající hardwarové konfiguraci o technologii vhodnou pro zpracování rozsáhlých datasetů o velikosti řádově desítek gigabajtů, s výhledem na urychlení výpočtů v příštích verzích knihovny. Nicméně pokud bychom museli čelit Big data problému, tak by bylo nutné použít odlišný přístup – na HPC[12] hardware zprovoznit distribuovanou infrastrukturu např. na bázi Apache Hadoop, nebo Apache Spark.

3.3 Reporty

V balíčku Output jsou implementované dva typy Reporterů:

- Standardní grafická okna, vytvářená knihovnou Matplotlib (možnost ručního zoomování a dalších operací), s textovým výstupem na konzoli.
- HTML výstup

HTML výstupy plní svůj účel docela dobře, ale po čase používání jsem objevil implementační omezení mnou provedeného řešení. Při analýze, mající za výstup mnoho set grafů, dojde k chybě exportu. Situace nastane pouze, pokud se zpracovává celý 1.5GB soubor naráz, zároveň je zvolena agregace na jízdy a více typů grafů naráz. Řešením by bylo ukládat vyexportované obrázky průběžně a HTML Reporter by je pouze zakomponoval do jednoho reportu. Nicméně, jak jsem psal výše (viz 2.4), prioritou je funkcionalita, proto jsem to více neřešil. Podotýkám, že při výstupu v podobě grafických oken by při takovém množství grafů nejspíš došla paměť u většiny pracovních stanic dneška.

3.4 Agregace podle dní

V tomto bodě jsem měl definováno zadání poněkud konkrétněji, než ve zbytku projektu. Pan Ing. Jiří Kubín, Ph.D. jako potenciální uživatel produktu mi sdělil, které výstupy považuje za důležité. Podle vlastního uvážení jsem doplnil některé typy

[12] High performance computing

vizualizací. Zkušenosti i zdrojový kód, získané v tomto bodě, jsem mohl využít dále (viz 3.5).

3.4.1 Konstruktor

Nejprve je do DataFramu přidán datový sloupec, který vznikne aplikací funkce __powerCat() na všechny řádky tabulky. Tato funkce každému sloupci přidělí informaci o tom, zdali v měřeném momentě docházelo k výrobě/spotřebě/nulové bilanci elektrické energie. Poté je nad daty zaveden nový index, kterým je sloupec s datem, podle kterého je DataFrame rozdělen na úseky.

```
self.df['powerCat'] = self.df['avg.P1[kW]'].apply(self.__powerCat)
df_date_index = self.df.set_index(keys = ["date"], inplace=False)
self.grouped = self.df_date_index.groupby(self.df_date_index.index, as_index = True)
```

Fragment kódu 2 - klíčové řádky vybrané z konstruktoru třídy DayLevel.

3.4.2 Numerické charakteristiky

Výstup pro jednotlivé dny může vypadat např. takto:

Max U = 816.60 V

Max I = 530.88 A

Max P = 369.91 KW

Max negative P = -270.11 KW

Real consuption E = 391.85 KWh

Total consuption E = 433.00 KWh

Total produce E = -41.14 KWh

Rekupered 10.50 %

Time of power on: 9:35:10 h:m:s

Položka rekupered zatím neodpovídá úplně skutečnosti, protože měřicí systém zatím nedovede rozpoznat, jestli je vyrobená energie vrácena do rozvodné sítě, nebo vyhozena v odporníku.

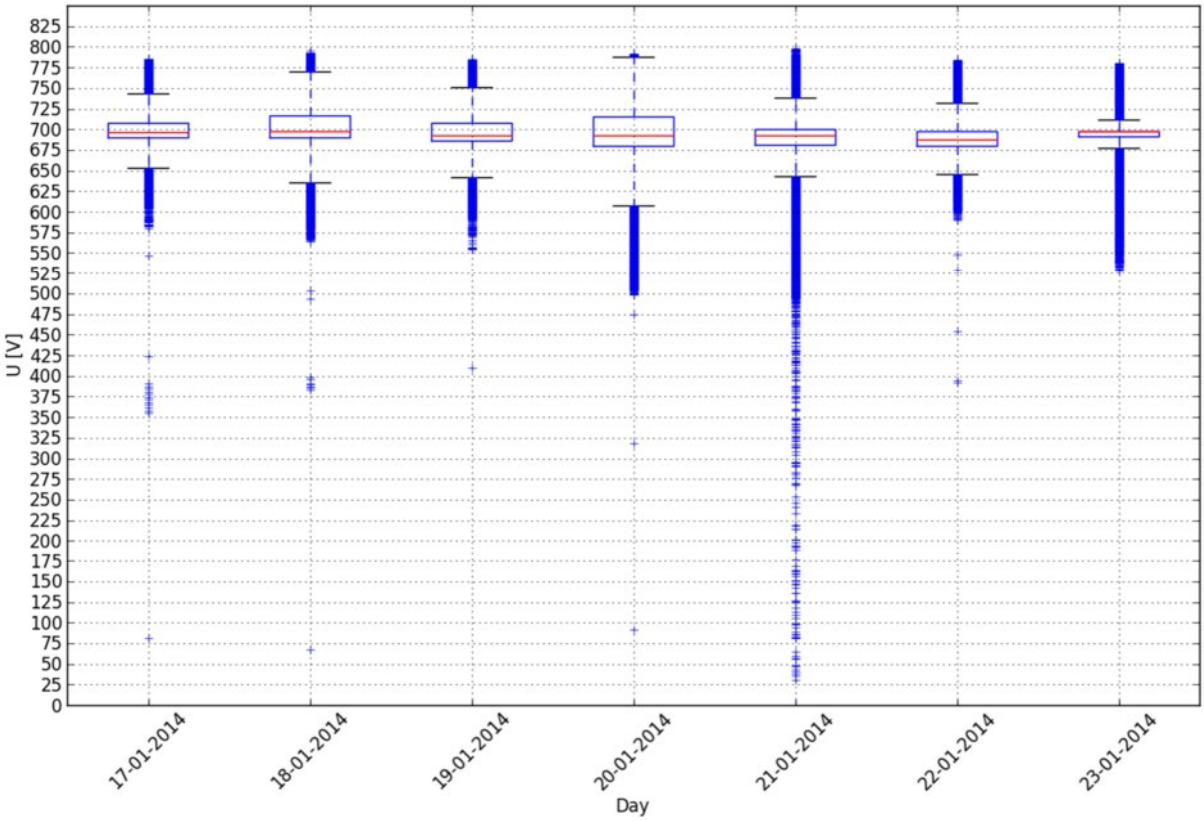
3.4.3 Krabicový graf

Krabicový graf (box plot) jsem použil jako souhrnnou charakteristiku pro popis kolísání napětí na přívodních svorkách tramvaje. Červená horizontální čára znázorňuje medián, horní a dolní strana obdélníku zobrazuje první a třetí kvartil. Úsečky nad a pod obdélníkem zobrazují interval, vypočítaný podle násobku mezikvartilového rozpětí:

$$\text{interval} = \text{whis} \times (Q_3 - Q_1)$$

$$\text{whis} = 1.8$$

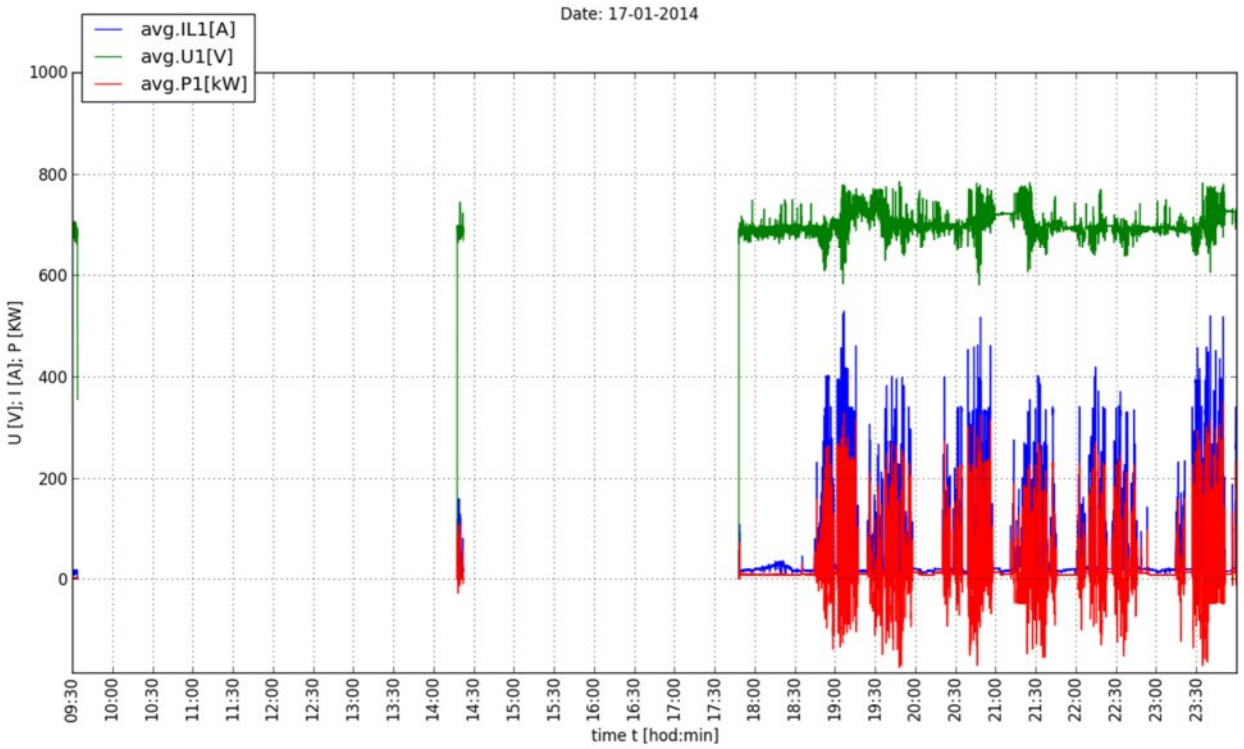
Hodnoty pod a nad / pod těmito přímkami jsou považovány za odlehlé pozorování.



Graf 4 - krabicový graf slouží k získání představy o kolísání napětí v rozvodné síti.

3.4.4 Spojnicový graf

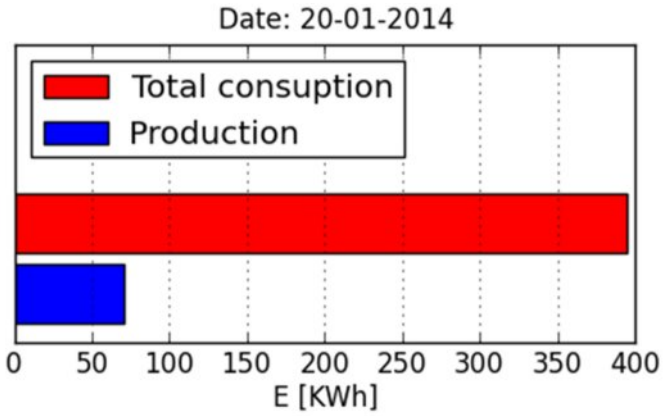
Podle doporučení (viz 3.4) jsem zobrazil všechny tři průběhy elektrických veličin do jednoho grafu. Pro popisky vodorovné osy jsem použil velice užitečnou funkci resample() z knihovny Pandas. Funkce umožňuje zvýšit i snížit vzorkovací frekvenci struktury DataFrame, nebo Series. Použil jsem podvzorkování, které umožňuje převzorkovat data tak, že v novém datasetu jsou kromě požadované periody i požadované časy, bez ohledu na přesný čas počátku záznamu. Díky tomu jsou na popiskách os zobrazeny celé hodiny, nikoliv pouze čas počátečního vzorku t a časy t + n×30 minut.



Graf 5 - průběh proudu, napětí a výkonu za jeden den.

Již jsem myslel, že mám tento typ výstupu hotový, ale všiml jsem si, že spočítané časy zapnutí tramvaje za den neodpovídají zobrazeným průběhům. Prověřil jsem možnost nestejné vzorkovací periody, jenže v tom to nebylo. Příčinou problému bylo, že měření neprobíhalo kontinuálně. Řešením bylo proložit časovou osu úseky, odpovídajícími době neaktivity měřicího přístroje.

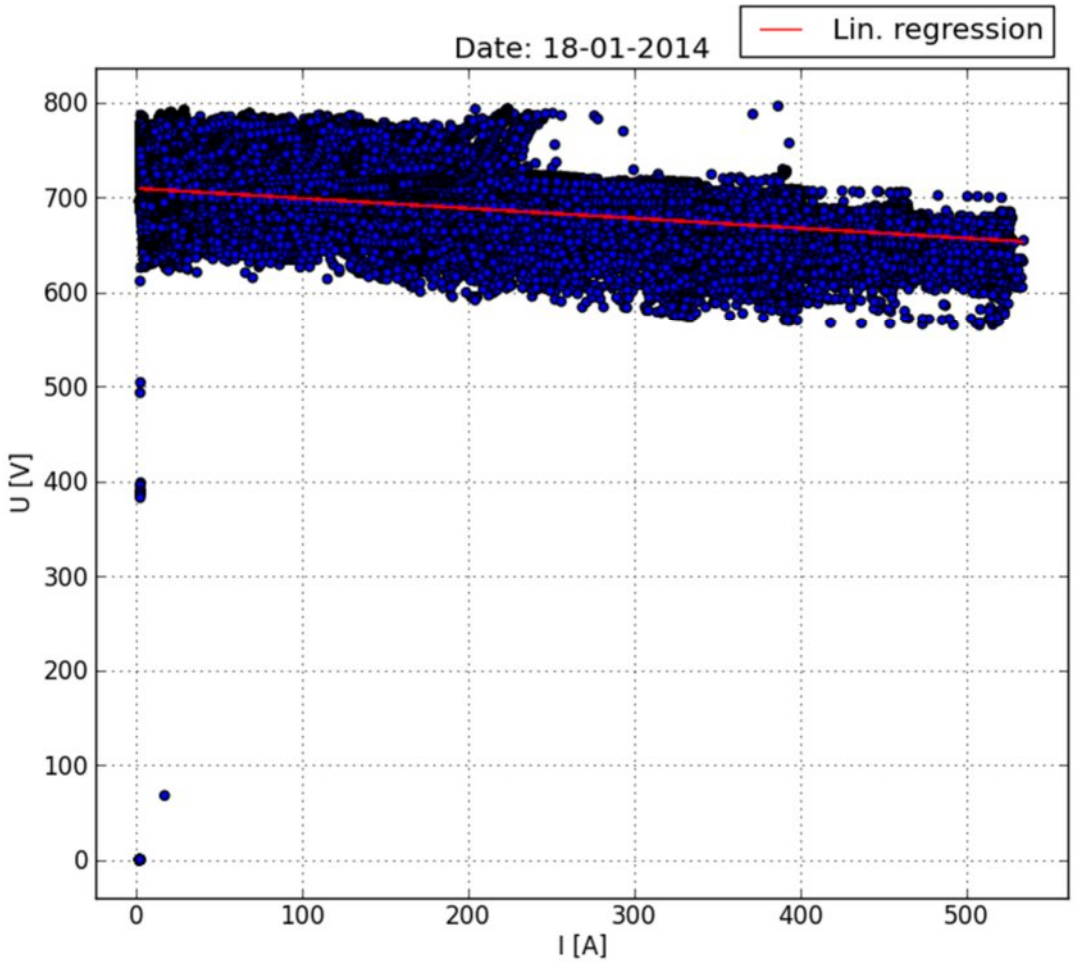
3.4.5 Sloupcový graf



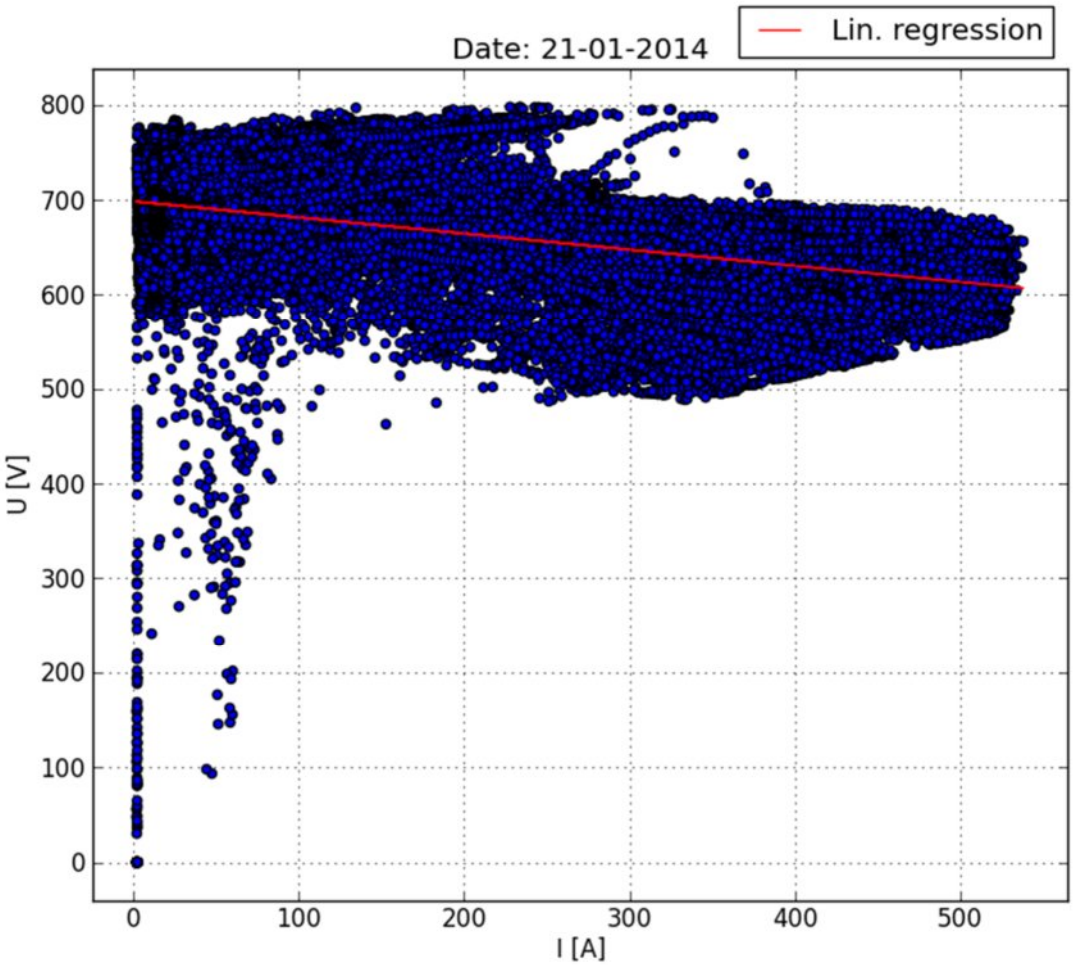
Graf 6 – Horizontální sloupcový graf znázorňuje přehledně bilanci výroby a spotřeby elektrické energie.

3.4.6 Korelační diagram

Korelační diagram, (scatter plot, XY graf) zobrazuje závislost velikosti napětí na odebíraném proudu. Na tomto typu grafu jsou poměrně dobře viditelná odlehlá pozorování (např. pokud je napětí příliš nízké). Mohou to být chyby měření, měření provedená po odpojení tramvaje od napájení, než se měřicí přístroj vypnul, anebo anomálie v rozvodné síti. Diagram jsem doplnil o přímku, znázorňující lineární regresi, vypočítanou pomocí knihovny Scikit-learn. Lineární regrese zobrazuje trendovou složku naměřených dat. Podrobněji se této funkční závislosti věnuji v kapitole 4.5.1.



Graf 7 – Den s malým množstvím anomálií.



Graf 8 – Den s velkým kolísáním napětí rozvodné sítě v blízkosti měřené tramvaje.

3.5 Agregace podle jízd

V tomto modulu fungují stejné výstupy jako při agregaci podle dní (viz 3.4). Jedná se o numerické charakteristiky (viz 3.4.2), zobrazení průběhu (viz 3.4.4), sloupcový graf (viz 3.4.5) a korelační diagram (viz 3.4.6). Kromě toho jsem experimentálně zvolil vhodnou veličinu pro rozdělení časových řad na jednotlivé jízdy. Na poslední chvíli jsem stihl i implementovat vybraný shlukovací algoritmus pro detekci, které úseky představují přestávky a které jízdy.

3.5.1 Provozní módy tramvaje

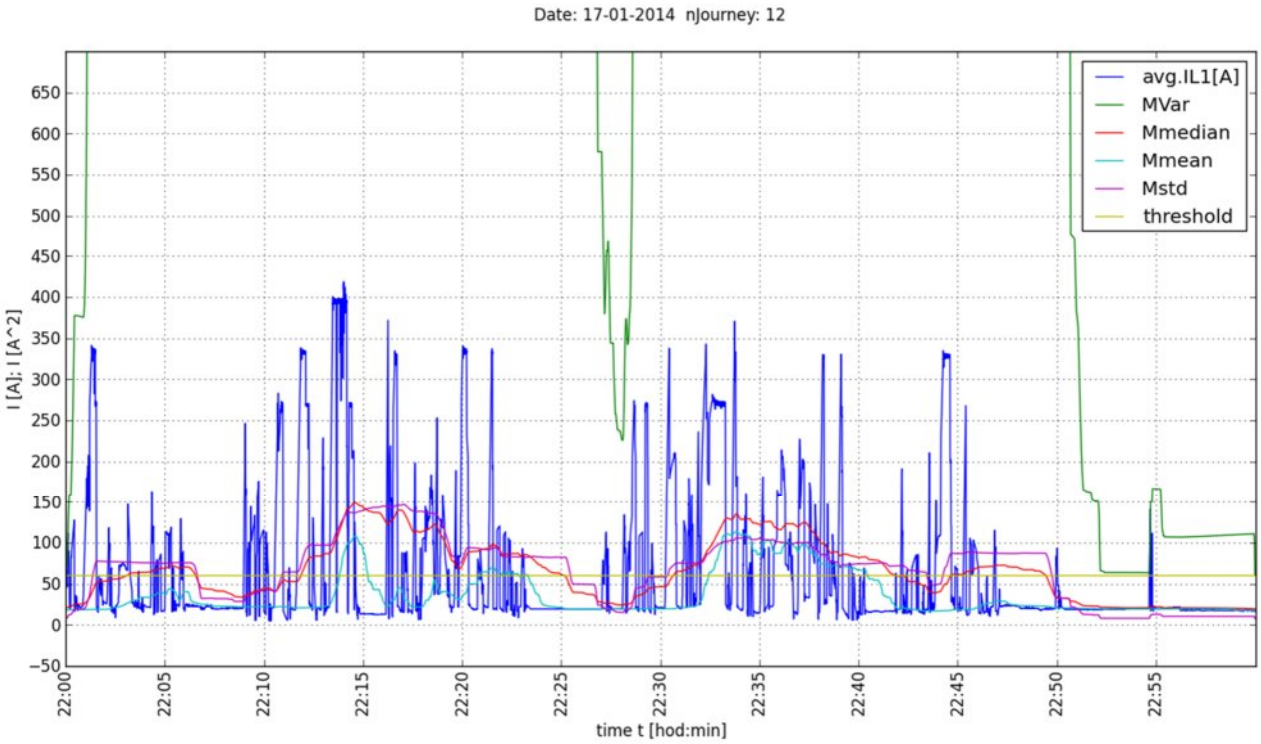
Nulový odběr – tramvaj odpojena od rozvodné sítě. Buďto záměrně, nebo z důvodu poruchy. Měřicí přístroj se vypíná.

Nízký odběr – je zapnuto n prvků z následujícího: řídicí počítač, vysílačka, světla, topení u řidiče, topení v prostoru pro pasažéry. Souprava nejspíš stojí na místě, buďto v depu, nebo na zastávce, nebo na křižovatce. V tomto stavu odběr nepřekročí cca 40A.

Vysoký odběr – jsou zapnuty spotřebiče, jako u nízkého odběru. Navíc je spuštěn trakční subsystém tramvaje. Tento stav je charakteristický velkým kolísáním velikosti proudu, čehož bude využito dále.

3.5.2 Identifikace jednotlivých jízd

Pro identifikaci jednotlivých jízd v datech může být použita prahová funkce, aplikovaná na nějakou veličinu, odvozená od změřeného průběhu proudu. Podstatnou komplikací je skutečnost, že jsem byl odkázán pouze na změřené hodnoty elektrických veličin, ze kterých lze s přijatelnou pravděpodobností (navíc za předpokladu, že naměřené hodnoty odpovídají realitě) stanovit pouze omezené množství stavů. Deník jízd není ve strojově čitelné podobě a jediné, co jsem kromě průběhu elektrických veličin znal, byly přibližné doby trvání zastávek.



Graf 9 - srovnání způsobů předzpracování průběhu proudu, před vstupem do prahové funkce. Klouzavý: rozptyl, medián, průměr.

Jako možné klouzavé veličiny jsem uvažoval průměr, medián, směrodatnou odchylku a rozptyl. Výši prahu jsem stanovil na 60A (viz 3.5.1). Dále bylo třeba experimentálně stanovit délku okna tak, aby nebyla zastavení na křižovatkách a v zastávkách detekována jako konec jízdy. Podle doby jízdy, zjištěné v jízdních řádech a podle opakujících se průběhů proudu, jsem se snažil oddělit zhruba třičtvrtěhodinové úseky, ohraničené pauzou dlouhou nejméně 5 minut. Samotný výpočet statistik probíhal pomocí funkcí obsažených v knihovně Pandas. Oproti úrovni

agregace na dny je propracovanější tvorba popisků osy x. Perioda popisků osy x je vypočítána podle délky zobrazovaného úseku (u dní to je napevno 30 minut).

Zvolil jsem klouzavý rozptyl s délkou okna t = 5 minut. Výhodou je, že strmě reaguje na rozjezd a je odolný proti kratším zastavením. Amplituda jde daleko nad horní mez osy y, protože je rozměr této veličiny druhá mocnina proudu. Směrodatnou odchylku[13] jsem do srovnání umístil až v době, kdy jsem byl rozhodnut pro použití rozptylu. Umístil jsem ji tam proto, abych mohl lépe srovnat průběh s veličinami, které nebyly umocněny na druhou. Sice reaguje o něco strměji než klouzavý průměr a medián, ale zas o tolik se neliší. Proto by mohl být zrovna tak použit klouzavý medián umocněný na druhou. Nevýhodou mého algoritmu přiřazení čísla úseku k datům je, že po konci jízdy vyhodnotí pauzu až se zpožděním. Kromě toho se stává, že déletrvající pauzu rozdělí na více úseků. Alternativní přístup by byl použít klasifikační algoritmus (viz Příloha A).

Podle experimentů se shlukovacími algoritmy (viz Kapitola 4) jsem nakonec do aplikace zakomponoval algoritmy K-means, MeanShift, DBSCAN. První jmenovaný může pracovat s více daty, takže pomocí něj lze klasifikovat provozní režimy v U(I) závislosti. Druhý a třetí algoritmus má kvadratickou paměťovou náročnost, takže ho lze použít pouze nad více agregovanými daty (např. rozpoznání pauza vs. jízda).

[13] Směrodatná odchylka je odmocnina z rozptylu.

4 Shluková analýza

Shluková analýza (cluster analysis) je široce používaná oblast strojového učení. Jedná se o metody učení bez učitele. Tyto metody mají za cíl pomocí mnohorozměrných statistických metod nalézt v dané množině objektů podmnožiny – shluky objektů – tak, aby si prvky v daném shluku byly navzájem podobné, ale nebyly si příliš podobné s objekty mimo shluk.

Cílem této fáze projektu bylo provést vyhodnocení použitelnosti vybraných shlukovačích algoritmů, obsažených v knihovně Scikit-learn na analyzovaných datech. Vzhledem ke komplikacím v počátečních stádiích projektu (viz Kapitola 2) jsem se tímto zajímavým tématem nemohl zabývat příliš do hloubky.

4.1 Základní rozdělení algoritmů

Shlukovací algoritmy dělíme na:

- Hierarchické (rozhodovací stromy)
- Nehiearachické

„Hierarchické shlukování je systém navzájem různých neprázdných podmnožin množiny, v němž průnikem každých dvou podmnožin je buď jedna z nich, nebo prázdná množina a v němž existuje alespoň jedna dvojice podmnožin, jejichž průnikem je jedna z nich.“ (11)

„Během shlukování probíhá sekvence vnořených rozkladů, která na jedné straně začíná triviálním rozkladem, kdy každý objekt dané množiny objektů tvoří jednoprvkový shluk, a na druhé straně končí triviálním rozkladem s jedním shlukem obsahujícím všechny objekty.“ (11)

„Nehierarchické shlukování je systém navzájem různých neprázdných podmnožin množiny, v němž průnikem každých dvou podmnožin není žádná z nich.“ (11)

4.2 Nehierarchické shlukování

Dále jsou rozebírány pouze metody nehierarchického shlukování. Výstupem použitých metod je množina vzájemně disjunktních množin (tříd v datech). Každá třída obsahuje nějaký typický exemplář, neboli etalon, který může, ale nemusí být prvkem původní množiny dat. Pro etalon, vypočítaný jako průměrná hodnota dat ze třídy, se používá označení centroid. Během shlukování probíhají následující úkony:

- Volba počtu podmnožin k.
- Výběr etalonů jednotlivých tříd.

- Vytvoření shluků kolem etalonů.

Při tvorbě shluků je třeba použít nějakou míru podobnosti. Bývají použity metriky založené na měření vzdálenosti bodů. Nejpoužívanější je euklidovská metrika.

4.3 Vyhodnocení modelu

Dalším krokem po klasifikaci je vyhodnocení výsledku. Pro různé metody bývají definovány různé účelové funkce (cost function). Dosazením klasifikovaných dat do této účelové funkce získáme tzv. skóre. Při učení s učitelem není problém určit procentuální úspěšnost klasifikace na testovacích datech (tzv. externí index). Protože u učení bez učitele správnou příslušnost do tříd neznáme, je třeba zvolit vhodnou číselnou charakteristiku, odvozenou z parametrů jednotlivých nalezených shluků (tzv. interní index).

4.3.1 Cíle vyhodnocení

Cílem takového vyhodnocení může být (12):

- Zjištění, jak dobře odpovídají nalezené shluky analyzovaným datům.
- Předcházení nalezení shluků v šumu.
- Porovnání shluků
- Porovnání shlukovacích algoritmů
- Nalezení správného počtu shluků.

4.3.2 Způsoby vyhodnocení

Vyhodnocení můžeme provést např. (12):

- Vizuálně
- Mírou soudržnosti – např. součet čtverců vzdáleností všech bodů od jim přiřazeného centroidu. (Dobré pro zjištění nejlepšího množství shluků, možnost zobrazit intervalové rozdělení kvadratických vzdáleností, pro jednotlivé vzorky pomocí histogramu.)
- Mírou separace – např. pro každé dvě třídy součet čtverců vzdáleností mezi všemi vrcholy bipartitního grafu.
- Silhouette coefficient – kombinuje míru soudržnosti a míru separace.

4.3.3 Silhouette coefficient

Pro kvantitativní vyhodnocení algoritmů ze Scikit-learn a nastavení jejich parametrů jsem použil Silhouette coefficient, zejména proto, že je v balíku již obsažen. Čím více se skóre blíží číslu 1, tím lépe jsou clustery definovány (13). Vzorec pro i-tý bod:

$$s_i = \frac{b - a}{\max(a, b)}$$

a … střední vzdálenost mezi vzorkem a zbytkem bodů stejné třídy.

b … střední vzdálenost vzorku a všech ostatních bodů dalšího nejbližšího clusteru.

Výsledná hodnota indexu je vypočítána jako průměr hodnot pro jednotlivé body.

4.4 Výběr příznaků

Požadavky na příznaky jsou: získatelnost, reprezentativnost, diskriminativnost, nekorelovanost (zdroj: materiály k předmětu ITE/MRK). Existují metody pro matematické vyhodnocení vybraných příznaků. Pro výběr konkrétních veličin pro konkrétní aplikaci však neexistuje obecný postup. Zpracovatel buďto využije své intuice založené na zkušenostech, nebo získá potřebné know-how vyhledáváním ve veřejných zdrojích, či koupí, pokud jsou požadované informace předmětem duševního vlastnictví.

Pro testování algoritmů jsem použil agregovaná data za jednotlivé jízdy. Jako kandidáty jsem použil: čas jízdy v sekundách, celková spotřeba energie, celková výroba energie, rozptyl proudu. Jak je vidět z tabulky, všechny atributy jsou silně korelované. Přesné hodnoty se mohou lišit podle načtených dat, ale odchylky nejsou příliš veliké. Záporné znaménko mezi výrobou energie a ostatními atributy je způsobeno tím, že hodnoty výroby mají záporné znaménko. Čím větší množství vyrobené energie, tím větší záporné číslo. Testy algoritmů jsem provedl na dvourozměrném příznakovém vektoru – výroba a spotřeba energie. Pro praktickou klasifikaci úseků jsem pak použil pouze množství spotřebované energie. Použití sofistikovanějšího přístupu při výběru příznaku jsem v této situaci nepovažoval za nezbytné.

Tabulka 3 – hodnoty pearsonova korelačního koeficientu mezi jednotlivými kandidáty na příznak.

	timeSec	totalConsuption	totalProduce	Variance
timeSec	1.000000	0.928039	0.908094	0.786092
totalConsuption	0.928039	1.000000	-0.893308	0.917086
totalProduce	-0.908094	-0.893308	1.000000	-0.805313
Variance	0.786092	0.917086	-0.805313	1.000000

4.5 Implementace v Scikit-learn

Pro představu o možnostech jednotlivých algoritmů jsem exportoval souhrnné hodnoty výroby a spotřeby elektrické energie do souboru v NumPy formátu. Poté jsem na upravených ukázkových skriptech z dokumentace ke knihovně Scikit-learn provedl

klasifikaci těchto dat metodami: K-means, AffinityPropagation, DBSCAN, MeanShift. Zkušební data měla rozměr 936 × 2 příznaky.

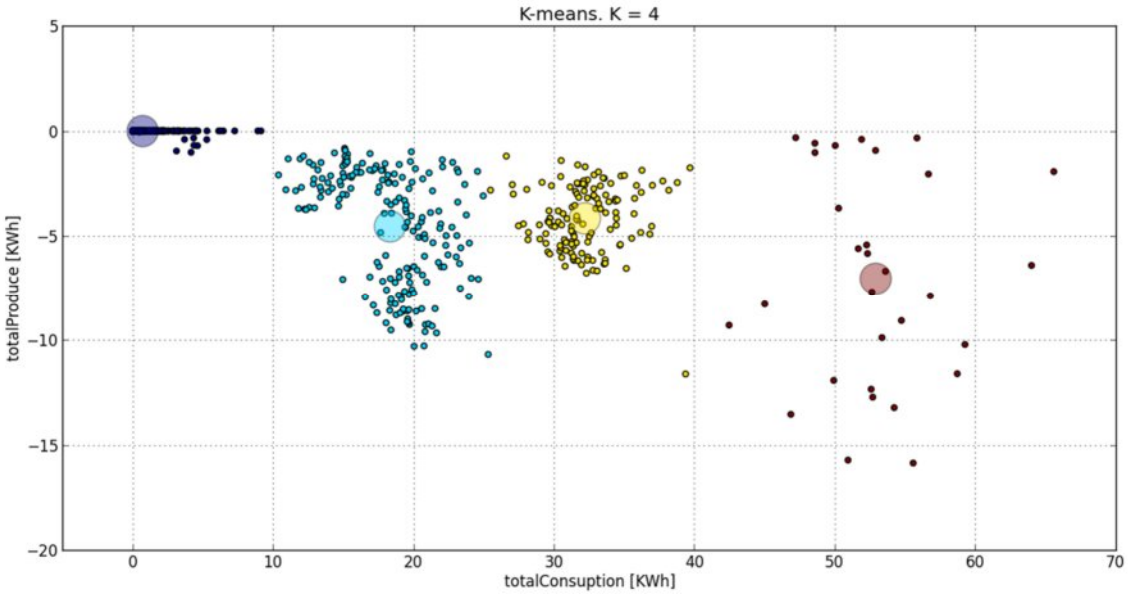
4.5.1 K-means

Jedná se pravděpodobně o nejznámější shlukovací algoritmus. Jako vstupní parametr se musí zadat požadovaný počet shluků k. Nejlepší počet shluků k můžeme odhadnout vizuálně, anebo zkoušíme různá n=1,2,… a podle výsledku shlukování vybereme nejlepší. Průběh algoritmu:

- Vybere náhodné body jako prvotní odhad centroidu.
- Každému prvku přiřadí nejbližší centroid.
- Pro každou skupinu vypočte nový centroid, aby měl nejmenší vzdálenost ke všem bodům třídy.
- Vyhodnotí celkové kritérium (interní index). Pokud se jeho hodnota liší od předchozí hodnoty o méně než ε, tak se ukončí, jinak návrat na krok 2.

Výsledek klasifikace se může lišit podle výběru počátečního centroidu. Průměrná časová složitost je dle dokumentace (14) O(k × n × T); tzn. asymptoticky O(n). Časová složitost v nejhorším případě: O(n^(k+2/p)).

k … počet tříd, n… počet prvků, T … počet iterací, p … počet příznaků



Graf 10 - výstup algoritmu K-means.

Nízká časová náročnost činí algoritmus použitelným i pro větší datasety. Implementace ve Scikitu navíc podporuje paralelní běh. Pro velmi rozsáhlé datasety je

pak určena modifikace algoritmu pojmenovaná Mini Batch K-Means. Více v (13). Nejlepší výsledek byl dosažen pro k = 4. Silhouette Coefficient = 0.823. Klasifikace trvala t = 0.122 s.

4.5.2 AffinityPropagation

U algoritmu AffinityPropagation jsou všechny datové body brány jako potenciální etalony. Počet tříd není třeba zadávat, je však ovlivněn zadávaným parametrem *preference*. Příliš vysoká hodnota *preference* vede k příliš velkému počtu nalezených tříd. Používá se například pro klasifikaci: síťového provozu, skupin opic, nalezení zajímavých bodů v obrazovém signálu, skupinový pohyb ve videu atd. (15).

Probíhá iterativní výměna zpráv mezi datovými body, dokud se neobjeví dobrá sada typických exemplářů a shluků (15).

Availability: zprávy zasílané od datových bodů – kandidátů na etalony, datovým bodům – potenciálním členům stejného shluku. Indikuje, jak vhodným kandidátem na etalon je daný bod.

Responsibility: zprávy zaslané od potenciálních členů shluku, kandidátům na etalon. Indikuje vhodnost zařazení bodu do daného shluku.

Similarity: s(i,j) je míra použitelnosti bodu j být etalonem pro bod i. Běžně se používá negativní hodnota euklidovské vzdálenosti.

Preference: parametr i-tého bodu p(i) je apriorní předpoklad použitelnosti jako etalon. Může být nastaveno globálně (sdílená hodnota mezi všemi body), nebo pro každý bod zvlášť. Vysoké hodnoty způsobí, že bude nalezeno mnoho etalonů.

Iterace: v každé iteraci je vypočítáno nové *responsibility* na základě aktuálního *availability*, *similarity* a *preference*. Poté jsou vypočítány *availability* podle aktualizovaných *responsibility*.

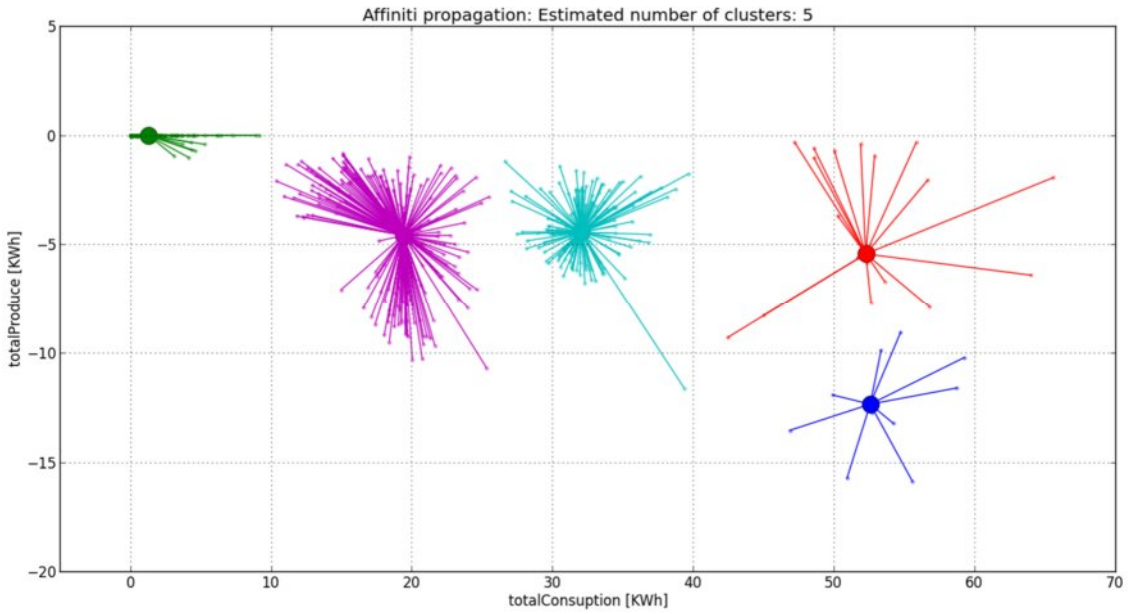
Konvergence algoritmu: Iterace jsou dokončeny, když se etalony a přiřazené body ve dvou iteracích po sobě nezmění.

Paměťové a časové požadavky jsou lineární ku počtu podobností, což je $O(N^2)$ pro množinu všech bodů, ale mnohem méně, pokud je množina podobností řídká. Pokud by byl algoritmus příliš časově nebo paměťově náročný, existuje verze pojmenovaná "leveraged affinity propagation" která vzorkuje množinu potenciálních podobností mezi dvojicemi bodů a provádí několik průchodů algoritmu na řídké množině podobností. Iteračně vylepšuje výběr vzorků (15).

Při testování implementace algoritmu ve Scikit-learn jsem použil globální hodnotu *preference* pro všechny body. Hodnotu parametru jsem zjistil iteračním

algoritmem, který nastavoval čísla v intervalu <-800;1> s krokem 1 a porovnával výsledné skóre Silhouette coefficientu.

Na testovacích datech bylo dosaženo nejlepšího výsledku při *preference* = -565. Při tomto nastavení bylo nalezeno 5 tříd. Algoritmus dosáhl vysokého skóre 0.910, ale bylo to vykoupeno dlouhou dobou zpracování. Klasifikace dat o rozměru 936×2 probíhala cca 20 s. Pro paměťovou a časovou náročnost implementace algoritmu ve Scikit-learn jsem ho do aplikace nezahrnul.

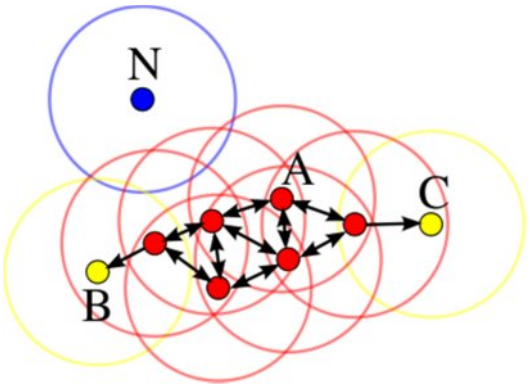


Graf 11 - výstup algoritmu AffinityPropagation.

4.5.3 DBSCAN

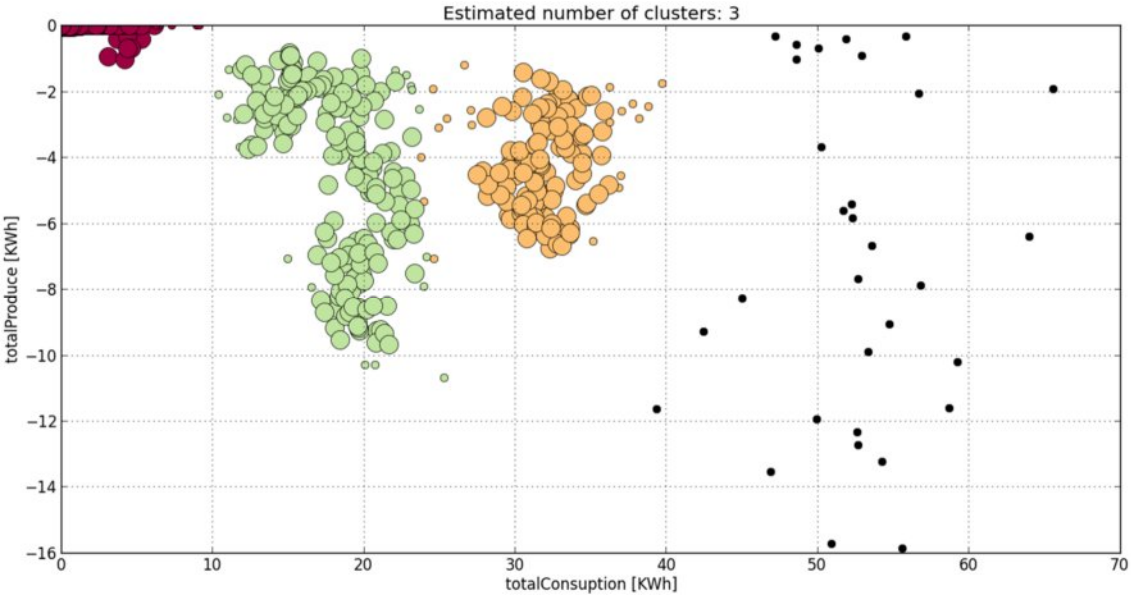
Celý název je: density-based spatial clustering of applications with noise (16). V překladu to znamená: na hustotě založený prostorový shlukovací algoritmus pro aplikace s šumem. Má dva parametry: *eps* a *min_samples*. První specifikuje poloměr okolí bodu, druhý minimální počet vzorku v okolí bodu. Pro každý bod p je spočítán počet bodů v okolí (hustota). Bod je poté označen jako (17):

- Core point: pokud má bod ve svém okolí více než *min_samples* bodů.
- Border point: pokud má v okolí méně než *min_samples* bodů, ale má v okolí Core point.
- Noise point: každý bod, který není ani Core point, ani Border point.



Obrázek 5 – algoritmus DBSCAN. Červeně Core point, žlutě Border point, modře Noise point (16).

Experimentálně jsem zvolil: *eps* = 4, *min_samples* = 50. Při slabším filtrování (nižší hodnota parametru *min_samples*), začalo vznikat v pravé oblasti grafu větší množství tříd, čítající pouze několik bodů. S tímto nastavením je naopak pravá oblast označena jako šum. Na testovacích datech běžel algoritmus t = 0.709 s. Hodnota Silhouette Coefficient = 0.819 je relativně vysoká. Nicméně koeficient nezohledňuje množství prvků označených jako šum.



Graf 12 – výstup algoritmu DBSCAN.

Při pokusu klasifikovat data v U(I) závislosti (viz 3.4.6) však algoritmus skončil výjimkou MemoryError. Podle

stackoverflow.com/questions/16381577/scikit-learn-dbscan-memory-usage

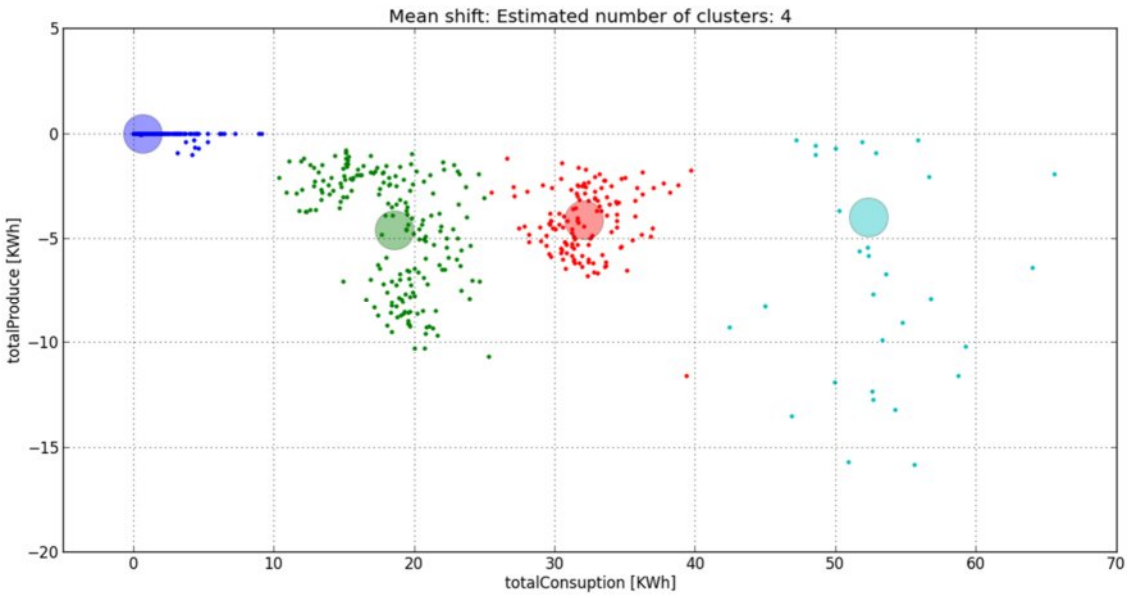
je problém v implementaci algoritmu ve Scikit-learn, protože se ukládá do paměti celá matice vzdáleností mezi body. Pro 100000 bodů to je $8\ B \times 10^{10} = 80GB$. Podle výše zmíněného zdroje dobrá implementace využívá hierarchický index a vystačí si s O(n) paměti a O(n log n) času.

4.5.4 MeanShift

Algoritmus MeanShift vyhledává lokální maxima hustoty bodů. Počet shluků se nemusí zadávat, je zjištěn algoritmem.

Časová složitost implementace ve Scikit-learn je asymptoticky O(n log(n)) pro nízké dimenze a $O(n^2)$ pro vyšší dimenze (18). Paměťová složitost v dokumentaci uvedena není. Na stroji s 4 GB RAM docházelo k výjimce MemoryError při pokusu klasifikovat více než cca 15000 bodů. Podle výpočtu $8 \times 15000^2 \div 1024^3 \approx 1.7GB$ odhaduji, že je paměťová složitost stejně jako u algoritmu DBSCAN $O(n^2)$.

Test na agregovaných datech skončil v čase t = 0.133. Silhouette Coefficient = 0.823



Graf 13 - výstup algoritmu MeanShift.

4.5.5 Vyhodnocení testu algoritmů

Pro rozsáhlejší datasety se ukázal jako jediný z testovaných algoritmů použitelný K-means. Zbylé algoritmy mohou dávat lepší výsledky, ale šly použít pouze na agregovaná data. Nejlepší shluky umí najít AffinityPropagation, za cenu dlouhé doby výpočtu. Indícií k výběru správného algoritmu ze Scikit-learn mohou být přílohy B a C.

Po zahrnutí klasifikačních algoritmů do aplikace (viz 3.5) jsem pomocí příznaku celková spotřeba energie za jízdu (vodorovná osa při testu v předchozím oddílu 4.5) klasifikoval jednotlivé úseky. Všechny algoritmy detekovaly první tři třídy na testovacích datech podobně:

- Zastavení maximálně s krátkou popojižďkou.
- Jízda s nižší spotřebou
- Jízda s vyšší spotřebou

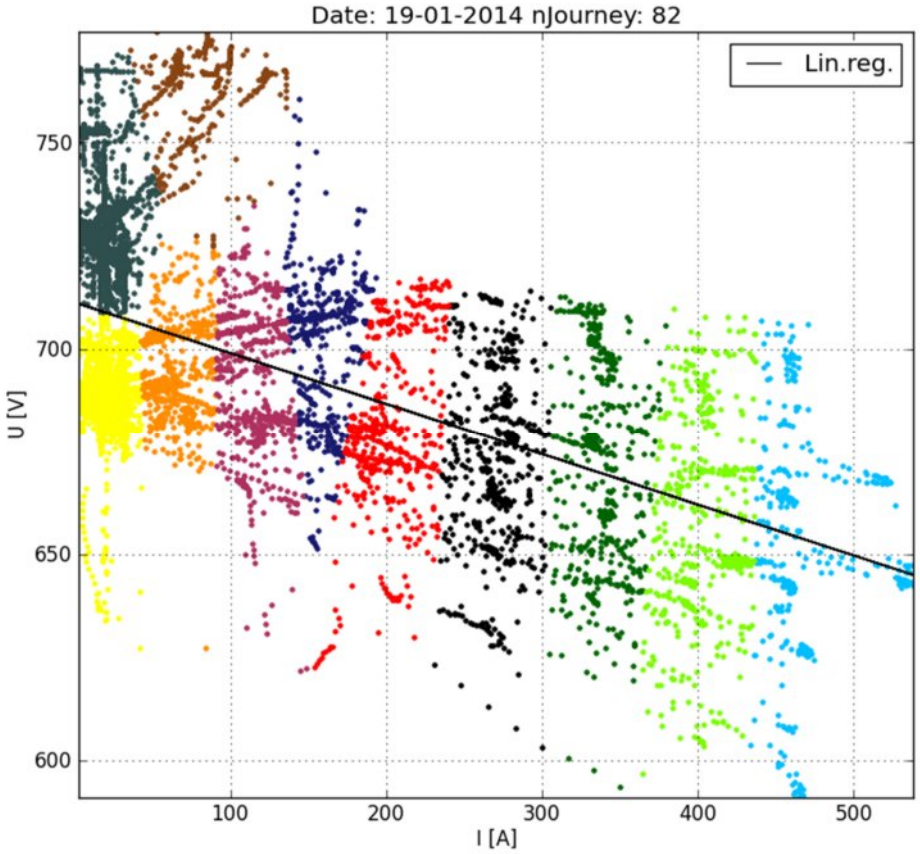
Body v pravé části zobrazené oblasti jsou buďto nestandardně dlouhé jízdy, kdy bylo spotřebováno více než 40 KWh energie, anebo dvě jízdy, mezi kterými byla krátká pauza, takže nebyly algoritmem (viz 3.5.2) rozděleny na dvě části. Zjistil jsem, že se mezi úseky „jízda“ trvající zhruba 45 minut vyskytují dvě třídy, kde průměrná spotřeba jedné je zhruba dvojnásobná oproti druhé. Možným vysvětlením by bylo, že se jízdy lišily počtem vozů v soupravě.

Při další práci s výsledky klasifikace je třeba uvažovat číselné charakteristiky centroidu. Identifikátor čísla třídy se může mezi průchody jednotlivých algoritmů lišit. Pro kvantitativní vyhodnocení přesnosti klasifikace by bylo nutné větší množství úseků označit ručně.

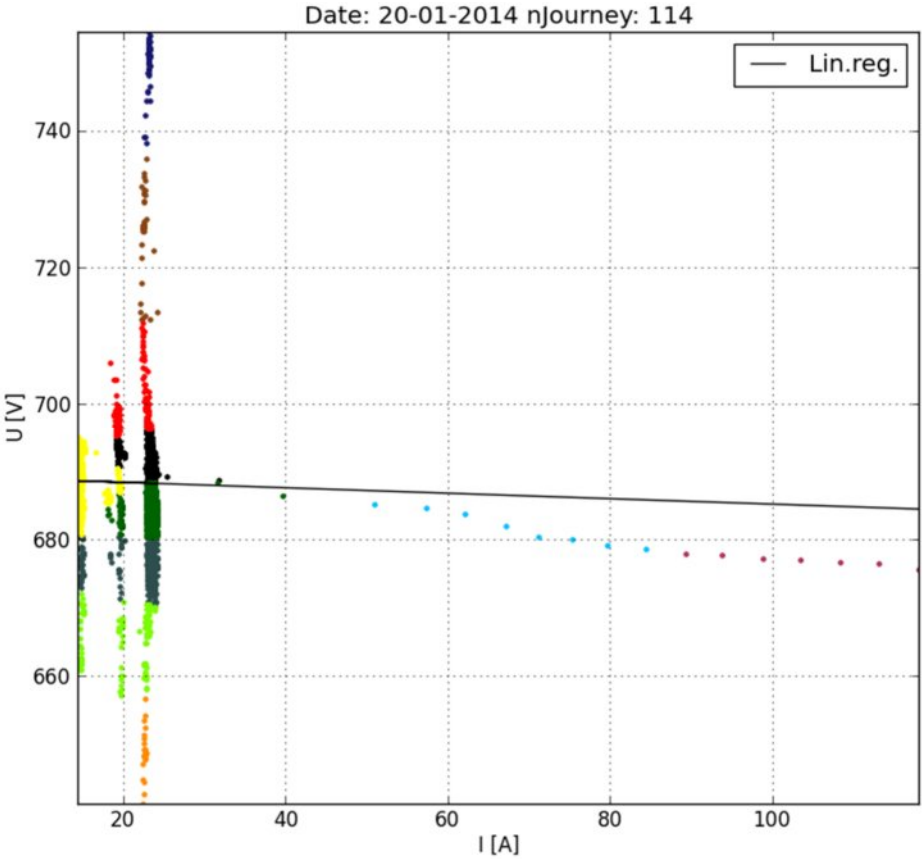
4.6 Klasifikace provozních režimů

Datových bodů v jednotlivých jízdách bylo tolik, že jsem použil pouze algoritmus K-means pro k = 11. Při vzorkovací periodě t = 200 ms je výstupem měření n = 3600 × 5 = 18000 vzorků za hodinu. Stropem algoritmů s paměťovou složitostí $O(n^2)$ je na počítači s Windows7 64b vybaveném 4GB RAM cca 15000 vzorků (viz 4.5.4).

Z grafu jízdy (viz Graf 14) je vidět známá závislost, kde se stoupajícím proudem klesá napětí v síti. Na grafu „pauzy“ (viz Graf 15) je zase vidět že při ustáleném odběru proudu je kolísání v rozvodné síti způsobeno jinými tramvajemi.



Graf 14 - klasifikace provozních režimů tramvaje při úseku „jízda“.



Graf 15 - klasifikace provozních režimů při úseku „pauza“.

5 Závěr

Závěrečným krokem datové analýzy bývá prezentace „příběhu“ analyzovaných dat. Pomocí explorační analýzy jsem získal základní představu o chování měřených veličin. Shluková analýza poodhalila některé typické stavy spotřeby energie při jednotlivých jízdách a provozní režimy elektronické výbavy tramvaje. Pro lepší analýzy by byly potřeba upřesňující informace (detekce otevírání dveří, strojově čitelný deník jízd, lepší znalost problémové domény). V současné době může archivace těchto dat sloužit spíše k demonstraci technologií a je diskutabilní, zda je ekonomicky efektivní skladovat celý archiv měření, nikoliv pouze agregovaná data. Může to být způsobeno nedostatečnou znalostí problémové domény (elektrické trakce), ale v datech nebyly zjištěny žádné souvislosti, které by mohly být použity k podpoře rozhodování managementu dopravního podniku. Při analýze dat nestačí umět používat sadu nástrojů, klíčem k zajímavým výsledkům je pokládání správných otázek (19).

Softwarový systém, který jsem vytvořil, může ušetřit práci s analýzou dat, ale ke komerčnímu nasazení má ještě dost daleko. Jsem přesvědčen o tom, že rozhodnutí začít vývoj znovu od začátku, po uplynutí zhruba půlky vymezeného času bylo správné. Pokračováním projektu by mohlo být uvedení do stavu vhodného k produkčnímu nasazení. Případným pokračovatelům doporučuji vytvořit mezioborový řešitelský tým.

Bibliografie

1. **ERBEN, Lukáš.** Děrné štítky a holocaust. [Online] 2013. http://www.root.cz/clanky/derne-stitky-a-holocaust-1/.

2. **Provost, Foster a Fawcett, Tom.** *Data Science for Business.* Sebastopol, CA : O'Reilly Media, Inc., 2013. ISBN: 978-1-449-36132-7.

3. **MATSUMURA, Miko.** Data Science Is Dead. [Online] 2014. http://news.dice.com/2014/03/05/data-science-is-dead/.

4. **O'Neil, Cathy.** *On Being a Data Skeptic.* Sebastopol, CA : O'Reilly Media, Inc., 2013. ISBN: 9781491947241.

5. **SKALSKÁ, Hana.** *Data mining a klasifikační modely.* Hradec Králové : Gaudeamus, 2010. ISBN 978-807-4350-887.

6. **MCKINNEY, Wes.** *Python for Data Analysis.* Sebastopol, CA : O'Reilly Media, Inc., 2012. ISBN 978-1449319793..

7. **DVOŘÁK, Jiří.** *Expertní systémy.* Brno : VUT v Brně, 2004.

8. SciPy. *Wikipedia.* [Online] http://en.wikipedia.org/w/index.php?title=SciPy&oldid=602603443.

9. OLAP cube. *Wikipedia.* [Online] 2014. http://en.wikipedia.org/w/index.php?title=OLAP_cube&oldid=606832336.

10. **Kraval, Ilja.** *Analytické modelování informačních systémů pomocí UML v praxi.* Valašské Klobouky : Object Consulting s. r. o., 2010. ISBN: 978-80-254-6986-6.

11. **Kelbel, Jan a Šilhán, David.** *Center for Machine Perception.* [Online] http://cmp.felk.cvut.cz/cmp/courses/recognition/zapis_prednasky/zapis_02/13/shlukovani.pdf.

12. **Ruoming, Jin.** Cluster validation. *Kent State University.* [Online] www.cs.kent.edu/~jin/DM08/ClusterValidation.pdf.

13. scikit-learn developers. *Clustering.* [Online] 2013. http://scikit-learn.org/stable/modules/clustering.html.

14. KMeans. *scikit-learn developers.* [Online] http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html.

15. AFFINITY PROPAGATION FAQ. *Probabilistic and Statistical Inference Group.* [Online] University of Toronto, 2007. http://www.psi.toronto.edu/affinitypropagation/faq.html.

16. DBSCAN. *Wikipedia.* [Online] http://en.wikipedia.org/w/index.php?title=DBSCAN&oldid=607541077.

17. **Ranka, Sanjay.** Clustering. *CISE Department: Data Mining.* [Online] http://www.cise.ufl.edu/class/cis4930sp09dm/notes/dm5part4.pdf.

18. MeanShift. *scikit-learn developers.* [Online] http://scikit-learn.org/stable/modules/generated/sklearn.cluster.MeanShift.html#sklearn.cluster.MeanShift.

19. **Shron, Max.** *Thinking with Data.* Sebastopol, CA : O'Reilly Media, Inc., 2014. ISBN: 978-1-449-36293-5.

20. *Visualization and Classification of Power System Frequency Data Streams.* **Bank, Jason N., Omitaomu, Olufemi A. a Liu, Yilu.** Miami, FL : 2009 IEEE International Conference on Data Mining Workshops, 2009. ISBN 978-0-7695-3902-7 .

21. Scaling Predictive Analytics. *Gartner Research .* [Online] 2013. http://www.gartner.com/technology/media-products/newsletters/Rapid_I/issue2/client.html.

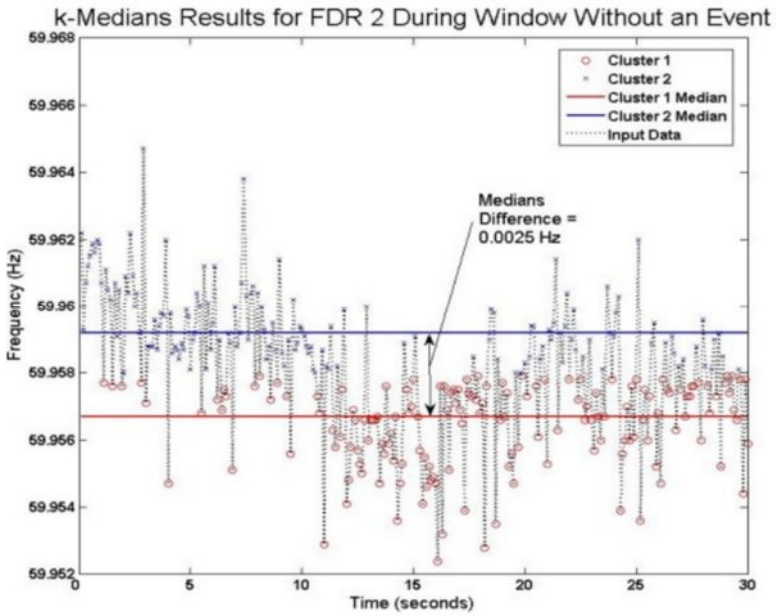
22. **Kart, Lisa, Heudecker, Nick a Buytendijk, Frank.** Survey Analysis: Big Data Adoption in 2013 Shows Substance Behind the Hype. [Online] Gartner Research, 2013. http://www.gartner.com/technology/media-products/newsletters/Rapid_I/issue2/gartner.html. G00255160.

23. **Hauck, Trent.** A Detailed Look at Pandas' Indexes. *O'Reilly community.* [Online] 2014. http://www.oreilly.com/pub/e/2921.

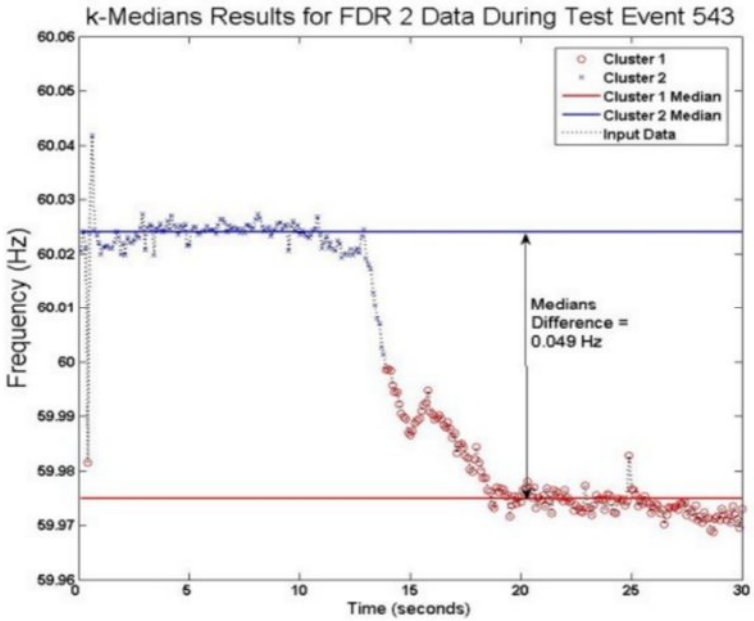
24. Choosing the right estimator. *scikit-learn developers.* [Online] http://scikit-learn.org/dev/tutorial/machine_learning_map/index.html.

Příloha A

Alternativní možnost detekce jednotlivých jízd by šlo realizovat úpravou postupu použitého při detekci poruch v rozvodné síti, jak popisuje článek (20). Článek popisuje detekci poruchových stavů v rozvodné síti pomocí online zpracování dat z měření frekvence. Na okně o délce 100 datových bodů je prováděna klasifikace pomocí algoritmu K-median pro K = 2. Důvod pro použití K-median místo K-Mean je jeho vyšší odolnost proti odlehlým pozorováním. Jak je vidět z obrázků, po klasifikaci dat do tříd se rozdíl mediánů tříd liší podle toho, zda došlo k poruše, či nikoli.

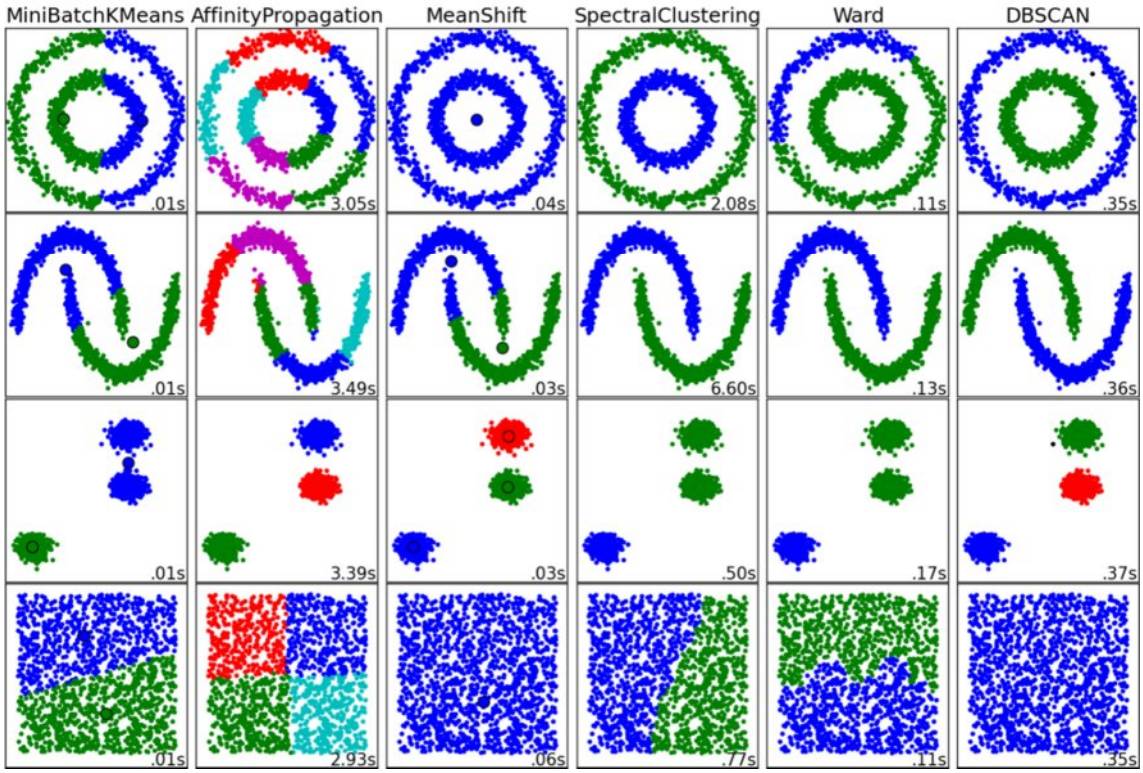


Obrázek 6 – nedošlo k poruše (20).



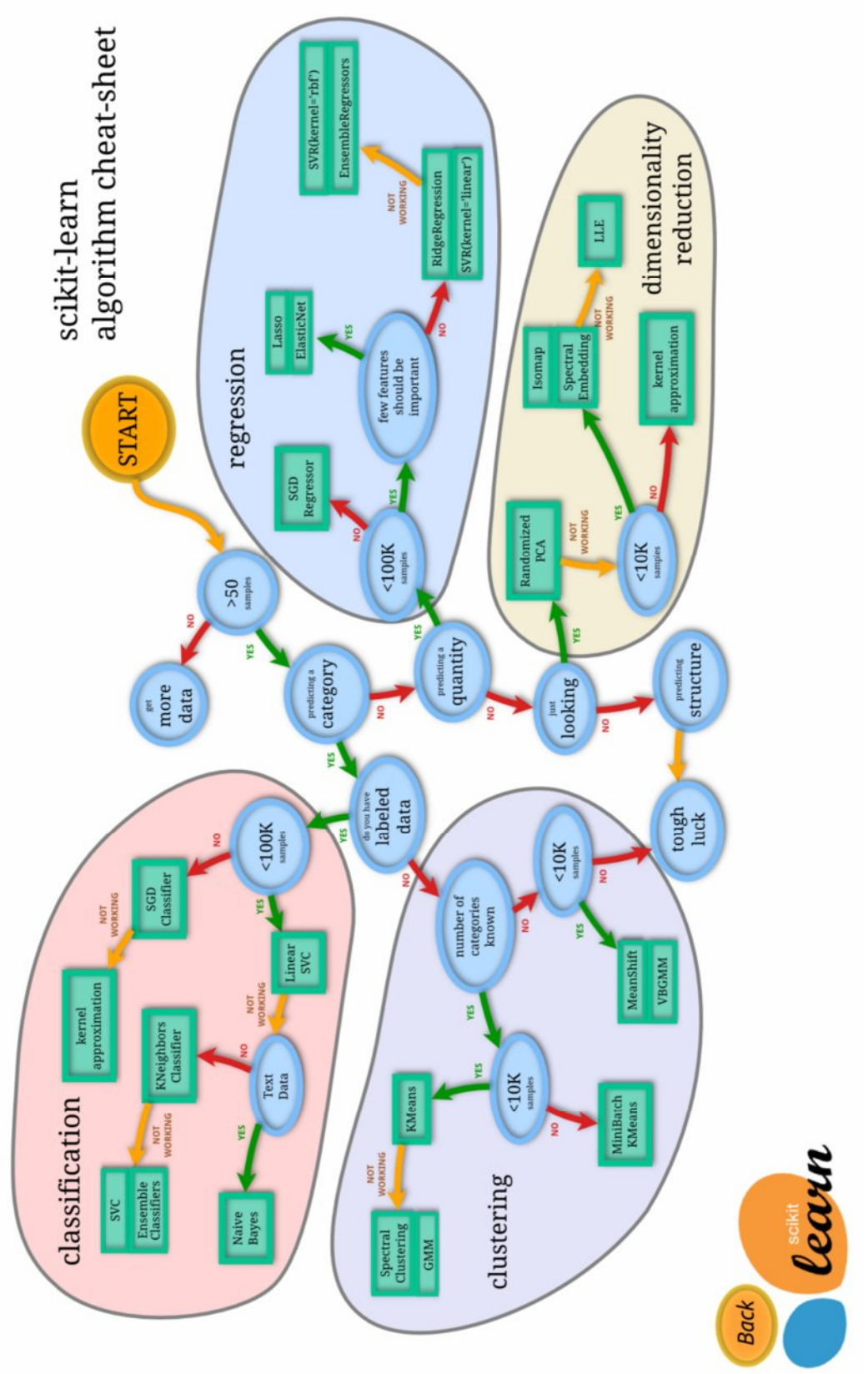
Obrázek 7 – došlo k poruše (20).

Příloha B



Obrázek 8 – srovnání shlukovacích algoritmů obsažených ve Scikit-learn (24).

Příloha C



Obrázek 9 - doporučené algoritmy pro jednotlivé typy úloh ve Scikit-learn (24).

Obsah přiloženého CD

├ BP_mikula_zprava.pdf	Textová část bakalářské práce.
├	
├──Clustering	Skripty se shlukovacími algoritmy
├──Feautures	Soubor s příznaky použitý při srovnání algoritmů
└──TramExpert	Aplikace